LAPORAN
PRAKTEK KERJA LAPANGAN

PENGEMBANGAN NARAWORKS :
HUMAN RESOURCE INFORMATION SYSTEM
DI PT NARAMEDIC

Sub Judul : Pengembangan Modul *Backend* Presensi dan *Human Resources Management* Terintegrasi dengan Sistem Penggajian

OKTA GABRIEL SINSAKU SINAGA
2207411017

PROGRAM STUDI TEKNIK IN*FORMA*TIKA
JURUSAN TEKNIK INFORMATIKA DAN KOM*PUT*ER
DEPOK
2025

HALAMAN PENGESAHAN

LAPORAN PRAKTIK KERJA LAPANGAN

a. Judul : PENGEMBANGAN MODUL *BACKEND* PRESENSI DAN HUMAN RESOURCE MANAGEMENT TERINTEGRASI DENGAN SISTEM PENGGAJIAN

b. Penyusun

i. Nama : Okta Gabriel Sinsaku Sinaga

ii. NIM : 2207411017

c. Program Studi : D4 Teknik In*form*atika

d. Jurusan : Teknik In*form*atika dan Komputer

e. Waktu Pelaksanaan : 18 Agustus 2025 s.d 12 Januari 2026

f. Tempat Pelaksanaan

i. Nama Perusahaan : PT. Naramedic Mitra Utama Solution

ii. Alamat Perusahaan : Blok *C*, Jl. Kemang Swatama No.27, Kalibaru, Kec. Cilodong, Kota Depok, Jawa Barat 16473

Depok, 06 Desember 2025

Pembimbing PNJ	Pembimbing Perusahaan
Dr. Anita Hidayati, S.Kom., M. Kom.	Hanief Mulia Ar Rosyid.
NIP. 197908032003122003	NRP/NIK

Mengesahkan,

KPS Teknik Informatika

Euis Oktavianti, S.S.I., M.T.I.

NIP. 198010272023212008

ABSTRAK

PT Naramedic Mitra Utama Solution merupakan perusahaan inovasi kesehatan yang memerlukan efisiensi tinggi dalam manajemen sumber daya manusia. Namun, proses pengelolaan data karyawan, absensi, dan penggajian sebelumnya masih menghadapi kendala dalam hal integrasi data dan validasi lokasi kerja yang akurat. **Rumusan masalah** dalam penelitian ini adalah bagaimana merancang dan mengimplementasikan sistem *backend* HRIS Naraworks yang mampu mengintegrasikan modul presensi berbasis lokasi (*geofencing*) dengan sistem penggajian secara otomatis dan aman.

Metode pengembangan perangkat lunak yang digunakan adalah Agile Scrum dengan siklus *sprint* dua mingguan. Pengembangan *backend* dibangun menggunakan *runtime* Node.js dengan *framework* Express.js, serta PostgreSQL sebagai sistem manajemen basis data relasional. Arsitektur sistem menerapkan pola *Modular Monolith* dengan struktur *Model-Service-Controller-Routes* untuk memastikan pemisahan tanggung jawab kode (*Separation of Concerns*). Pengujian sistem dilakukan menggunakan metode *Black Box Testing* melalui alat bantu Postman untuk memvalidasi fungsionalitas setiap *endpoint* API dan memastikan keamanan autentikasi berbasis *JSON Web Token* (JWT).

Hasil dari pelaksanaan magang ini adalah sebuah sistem *backend* HRIS yang terintegrasi. **Sebelum** adanya sistem ini, pencatatan kehadiran dan perhitungan lembur dilakukan secara manual yang rentan terhadap kesalahan manusia dan kecurangan lokasi. **Sesudah** implementasi, sistem mampu melakukan validasi kehadiran secara otomatis menggunakan algoritma Haversine dengan toleransi jarak (*geofencing*), melakukan perhitungan jam kerja serta lembur secara *real-time*, dan menyajikan data yang siap dikonsumsi oleh modul *payroll*. Infrastruktur yang digunakan, mulai dari laptop pengembangan MSI Prestige 14H hingga *deployment* menggunakan Docker di server VPS Hostinger, memastikan sistem berjalan stabil dengan respon API yang efisien.

Kata Kunci: *Backend*, Node.js, Express.js, PostgreSQL, *Geofencing*, Haversine, HRIS.

KATA PENGANTAR

Puji dan syukur penulis panjatkan *ke* hadirat Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan laporan Praktik Kerja Lapangan (PKL) ini. Laporan ini disusun sebagai salah satu syarat dalam menyelesaikan pendidikan Program Diploma Empat di Politeknik Negeri Jakarta.

Praktik Kerja Lapangan dilaksanakan di **PT. Naramedic Mitra Utama Solution**. Melalui kegiatan magang ini, penulis memperoleh manfaat berupa pemahaman penerapan pengembangan sistem informasi di dunia kerja, khususnya pada pengembangan modul *backend*, serta pengalaman bekerja dalam lingkungan profesional.

Selama pelaksanaan PKL, penulis menghadapi beberapa hambatan, seperti penyesuaian terhadap alur kerja industri dan pemahaman dokumentasi sistem. Hambatan tersebut dapat diatasi dengan bimbingan dan kerja sama yang baik dari berbagai pihak.

Penulis mengucapkan terima kasih kepada **Ibu Dr. Anita Hidayati, S.Kom., M.Kom.** selaku dosen pembimbing PKL, **Bapak Hanief Mulia Ar Rosyid** selaku pembimbing industri di PT. Naramedic Mitra Utama Solution, **Bapak Ang*git*o** selaku mentor, serta **keluarga dan teman-teman** yang telah memberikan dukungan selama pelaksanaan PKL.

Penulis menyadari laporan ini masih memiliki keterbatasan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan. Semoga laporan ini dapat memberikan manfaat bagi pembaca dan menjadi bekal bagi penulis dalam dunia kerja.

Depok, 20 Desember 2025

Okta Gabriel Sinsaku Sinaga

Daftar Isi

Daftar Gambar

Daftar Tabel

BAB I

PENDAHULUAN

Bab ini berisi uraian pendahuluan yang memberikan gambaran umum mengenai pelaksanaan Praktik Kerja Lapangan (PKL). Pembahasan diawali dengan latar belakang pengembangan sistem yang menjelaskan kondisi, permasalahan, serta kebutuhan perusahaan terhadap sistem informasi sumber daya manusia. Selanjutnya diuraikan ruang lingkup kegiatan pengembangan *backend* yang dilakukan selama kegiatan PKL, termasuk batasan pekerjaan yang ditetapkan. Pada bab ini juga disajikan informasi mengenai waktu dan tempat pelaksanaan PKL, sistem pelaksanaan pengembangan, serta tujuan dan kegunaan pengembangan sistem bagi institusi pendidikan dan perusahaan. Melalui penyajian bab pendahuluan ini, diharapkan diperoleh pemahaman awal mengenai konteks dan arah pembahasan pada bab-bab selanjutnya.

1.1 Latar Belakang

Proses pengelolaan sumber daya manusia *(Human Resource Management)* pada perusahaan modern menuntut sistem yang terintegrasi, akurat, dan mampu bekerja secara *real-time*. Dalam banyak kasus, proses administrasi seperti pencatatan presensi, pengelolaan lembur, dan perhitungan penggajian masih dilakukan secara manual. Metode ini memiliki beberapa kelemahan, seperti potensi kesalahan *input*, keterlambatan rekapitulasi data, duplikasi pencatatan, serta kesulitan integrasi antardivisi.

PT Naramedic Mitra Utama Solution sebagai perusahaan yang bergerak di bidang layanan kesehatan dan pengadaan alat medis membutuhkan sistem internal yang dapat meningkatkan efisiensi operasional. Salah satu kebutuhan utamanya adalah platform ***Human Resource Information System*** **(HRIS)** yang mampu:

a. Mencatat presensi harian pegawai secara terstruktur, termasuk presensi lokasi kerja dan kunjungan lapangan.
b. Mengelola data lembur berdasarkan waktu aktual dan parameter regulasi perusahaan.
c. Menyediakan perhitungan penggajian yang terintegrasi dengan data presensi, lembur, serta variabel operasional lainnya.
d. Menjamin keamanan data dengan autentikasi dan otorisasi yang sesuai *stand*ar industri.
e. Menghadirkan *API backend* yang stabil dan andal untuk digunakan aplikasi *mobile* dan web.

Berdasarkan kebutuhan tersebut, dikembangkanlah Naraworks, sebuah sistem HRIS internal yang berfokus pada digitalisasi presensi dan penggajian. Pada laporan ini, pembahasan difokuskan pada **pengembangan modul *backend***, meliputi:

a. Perancangan arsitektur *database* HRM dan *Payroll*.
b. Penyusunan *endpoint REST API* untuk presensi harian, presensi kunjungan, lembur, dan rekapitulasi data.
c. Penerapan logika bisnis perhitungan lembur dan penggajian.
d. Implementasi *middleware* autentikasi dan otorisasi.
e. Pengembangan mekanisme validasi data, *error handling*, dan integrasi modul.

Sistem ini dibangun menggunakan pendekatan *server-side* modern berbasis ***Express.js***, dengan tujuan menciptakan layanan *backend* yang cepat, aman, dan mudah dikembangkan lebih lanjut. Pengembangan Naraworks diharapkan dapat mengatasi permasalahan yang sebelumnya muncul pada proses manual, sekaligus menjadi fondasi bagi integrasi sistem HR yang lebih luas di masa depan.

1.2 **Ruang** Lingkup **Kegiatan**

Ruang lingkup pengembangan dalam proyek ini difokuskan pada perancangan dan implementasi **modul *backend Human Resource Information System* (HRIS)** Naraworks, khususnya pada bagian **presensi karyawan**, **pengelolaan data HRM**,

dan **integrasi dengan modul penggajian**. Ruang lingkup pekerjaan mencakup aspek teknis berikut:

a. Perancangan Sistem dan Basis Data
    1) Menganalisis kebutuhan data untuk modul HRM, presensi, lembur, cuti, kunjungan kerja, dan data pendukung *payroll*.
    2) Menyusun perancangan basis data menggunakan *Entity Relationship Diagram* (ERD).
    3) Merancang *skema database relasional* berbasis *PostgreSQL*.
    4) Melakukan implementasi *database* dalam lingkungan pengembangan berbasis *Docker*.

b. Pengembangan *API* dan Implementasi Logika Bisnis

Mendesain *struktur API* dan *endpoint REST* untuk seluruh modul presensi, seperti:

    1) Presensi harian
    2) Presensi lembur
    3) Presensi kunjungan kerja
    4) Data cuti
    5) Hari libur nasional
    6) Mengembangkan logika bisnis presensi dan integrasinya dengan HRM.
    7) Mengimplementasikan mekanisme validasi, *error handling*, dan *middleware* autentikasi serta otorisasi.
    8) Membuat penjadwalan otomatis menggunakan *cron job* untuk kebutuhan presensi tertentu.
    9) Melakukan integrasi data HRM–Presensi dengan *struktur* penggajian yang dikembangkan pada modul *payroll*.

c. Kolaborasi Teknis dalam Pengembangan Sistem
    1) Melakukan sinkronisasi *API* dengan modul *frontend* web dan *mobile*.
    2) Melakukan integrasi version control menggunakan *Git*Hub.
    3) Menjalankan proses pengembangan menggunakan *stand*ar praktik perangkat lunak seperti *code* review, *API* documentation, dan *stand*ar penulisan kode.

d. *Tools* dan Teknologi Pengembangan

1) *Backend*: *Node.js*, *Express.js*
2) *Database*: *PostgreSQL*, pgAdmin
3) *Development Tools*: *Visual Studio Code*, *Postman*
4) *Containerization*: *Docker*
5) Repository & CI/CD: *Git*Hub
6) Batasan Ruang Lingkup
7) Pengembangan tidak mencakup:
8) Implementasi antarmuka pengguna (*frontend*).
9) Pengembangan penuh modul *payroll* di luar integrasi data.
10) Aktivitas operasional HR non-teknis.
11) Pengembangan aplikasi *mobile* Naraworks.

1.3 Waktu dan Tempat Kegiatan

1.3.1 Identitas Instansi Tempat Magang

Berikut adalah informasi mengenai instansi tempat pelaksanaan kegiatan magang:

Nama Perusahaan	: PT Naramedic Mitra Utama Solution
Alamat Perusahaan	: Blok *C*, Jl. Kemang Swatama No.27, Kalibaru, Kec. Cilodong, Kota Depok, Jawa Barat 16473
Bidang Usaha	: Konsultan Perizinan dan Pengadaan Alat-Alat Medis (boleh tambahkan detail jika diperlukan)
Unit/Divisi Penempatan	: Divisi IT – Tim *Backend Development*
Sistem Kerja	: *Hybrid* (*Work From Anywhere* dan pertemuan *onsite* tertentu)

1.3.2 Periode Magang

Kegiatan magang dilaksanakan pada:

Tanggal Pelaksanaan	: 18 Agustus 2025 – 12 Januari 2026
Durasi Total	: 20 Minggu / 4 Bulan
Hari Kerja	: Senin–Sabtu
Jam Kerja	: Fleksibel sesuai kebijakan perusahaan. Senin-Jumat : Sistem Kerja Fleksibel (

Work From Anywhere)

Sabtu : 13.00 – 17.00 WIB

Seluruh rangkaian kegiatan dilaksanakan sesuai arahan pembimbing lapangan serta ketentuan perusahaan mitra.

1.3.3 Sistem Pelaksanaan Pengembangan

Pelaksanaan pengembangan sistem dilakukan menggunakan mekanisme kerja pengembangan perangkat lunak standar, meliputi:

1. Pengembangan Berbasis *Remote* dan *Onsite* (*Hybrid*)
   a. Pengembangan *backend* dilakukan melalui lingkungan kerja digital menggunakan *repository GitHub* dan *tools* pendukung.
   b. Koordinasi teknis dilakukan melalui media komunikasi internal perusahaan.
2. Pertemuan Teknis dan Review

   Pertemuan evaluasi dilakukan untuk:
   a. Review modul dan *API* yang telah dikembangkan
   b. Penyelarasan kebutuhan sistem HRIS
   c. Integrasi lintas modul (*frontend – backend – payroll*)
3. Kolaborasi Pengembangan Sistem

   Pengembangan dilakukan bersama tim internal yang terdiri dari:
   a. *Backend Developer*
   b. *Frontend Web Developer*
   c. *Mobile Developer*

Kolaborasi dilakukan melalui *repository*, komunikasi teknis, dan dokumentasi *API*.

1.4 Tujuan dan Kegunaan

Penyusunan laporan dan pengembangan sistem Naraworks ini didasarkan pada target pencapaian tertentu yang mencakup aspek teknis maupun fungsional. Berikut adalah rincian tujuan serta kegunaan dari pengembangan *backend* sistem tersebut:

1.4.1 Tujuan Pengembangan

Tujuan dari pengembangan *backend* sistem Naraworks ini adalah:

1. Tujuan Umum

   Mengembangkan *backend* HRIS yang mampu mengelola data kehadiran dan HRM secara terstruktur dan dapat diintegrasikan dengan sistem penggajian.

2. Tujuan Khusus

   a. Merancang dan membangun *struktur database* untuk modul presensi dan HRM.

   b. Mengembangkan *API* dan *endpoint REST* untuk presensi harian, lembur, kunjungan, cuti, dan hari libur.

   c. Menerapkan logika bisnis yang sesuai dengan kebutuhan perusahaan.

   d. Mengintegrasikan modul presensi dengan modul *payroll* sebagai dasar perhitungan gaji.

   e. Mengimplementasikan sistem keamanan melalui autentikasi dan otorisasi.

   f. Menghasilkan dokumentasi teknis yang dapat digunakan untuk pengembangan selanjutnya.

1.4.2 Kegunaan Pengembangan Sistem

Pengembangan sistem ini memberikan manfaat ber*up*a peningkatan kemampuan mahasiswa dalam merancang arsitektur *backend*, membangun *RESTful API*, mengimplementasikan manajemen *database* terstruktur, serta melakukan integrasi modul presensi dan *payroll* secara terpadu. Bagi perusahaan, sistem ini menyediakan layanan *backend* HRIS yang dapat langsung digunakan untuk mendukung proses operasional seperti pencatatan presensi, otomatisasi jam kerja, pengolahan data lembur, serta penyediaan data terstruktur yang siap diintegrasikan dengan modul lain. Selain itu, dokumen teknis yang dihasilkan dapat dijadikan acuan untuk pengembangan dan skalabilitas sistem di masa mendatang.

BAB II

LANDASAN TEORI

Bab ini memuat landasan teori yang digunakan sebagai dasar dalam pelaksanaan dan pengembangan sistem pada kegiatan Praktik Kerja Lapangan. Pembahasan mencakup konsep-konsep dasar terkait manajemen sumber daya manusia, sistem informasi, serta proses bisnis presensi dan pengelolaan data kepegawaian. Selain itu, pada bab ini diuraikan teori-teori pendukung mengenai teknologi *backend* yang digunakan, meliputi arsitektur aplikasi, basis data, *containerization, version control*, serta mekanisme komunikasi data. Bab ini juga membahas metodologi pengembangan perangkat lunak yang diterapkan dan konsep keamanan sistem sebagai upaya menjaga integritas serta kerahasiaan data. Penyajian landasan teori ini bertujuan memberikan kerangka konseptual yang mendukung analisis dan implementasi sistem pada bab-bab selanjutnya.

2.1 Proses Bisnis HR dan Presensi

Bagian ini membahas konsep dasar yang berkaitan dengan proses bisnis *Human Resource Management* (HRM), sistem HRIS, dan teori presensi elektronik yang menjadi dasar pengembangan sistem.

2.1.1 Pengertian *Human Resource Management* (HRM)

Human Resource Management (HRM) atau manajemen sumber daya manusia merupakan serangkaian proses yang mencakup perencanaan, pengelolaan, dan pengembangan sumber daya manusia dalam suatu organisasi. Menurut Eka Rakhmat Kabul (2024), teknologi dalam HRM telah digunakan untuk mendukung proses seperti rekrutmen, pelatihan, pengembangan, serta pemantauan kinerja karyawan melalui berbagai platform seperti HR *Information System*s (HRIS) dan *Applicant Tracking System*s (ATS).

Kabul (2024) menjelaskan:

"Teknologi HRM merujuk pada sistem yang memanfaatkan teknologi informasi *untuk meningkatkan efisiensi dan efektivitas pengelolaan sumber daya manusia, termasuk proses seperti pelatihan, pengembangan, dan pemantauan kinerja karyawan."*

Dengan adanya digitalisasi, HRM modern tidak hanya berkaitan dengan pengelolaan SDM, tetapi juga melibatkan otomatisasi proses administrasi seperti presensi, cuti, lembur, serta pengelolaan data karyawan.

2.1.2 Komponen HRM yang Relevan dengan Sistem

Berdasarkan analisis Sistem Integrasi PT Angkasa Raya Steel (SISTAR) dalam jurnal Manajemen (2025), komponen HRM yang relevan dengan pengembangan sistem meliputi:

a. Data Karyawan

Meliputi data kehadiran, izin, cuti, lembur, pergantian shift, serta informasi kepegawaian lainnya. SISTAR digambarkan sebagai sistem terpadu yang mendukung pengelolaan data tersebut untuk kebutuhan analisis dan pengambilan keputusan HR.

b. *Attendance* (Presensi)

Merupakan komponen utama dalam pengelolaan HR. Sistem presensi memungkinkan pencatatan waktu hadir secara *real-time*, mengurangi kesalahan pencatatan manual, dan membantu memantau kedisiplinan. Sistem pelacakan kehadiran berbasis lokasi merupakan salah satu pengembangan komprehensif untuk meningkatkan akurasi data tersebut (Gupta et al., 2025).

c. Cuti dan Izin

Proses administrasi cuti dan izin dikelola melalui pengisian *form*ulir digital yang mencakup tanggal, jenis izin, keperluan, serta status persetujuan.

d. Lembur (*Overtime*)

Pencatatan lembur dilakukan dengan menentukan tanggal, durasi, dan uraian pekerjaan. Sistem membantu HRD melakukan rekapitulasi lembur secara cepat dan akurat.

e. *Payroll* (Penggajian)

Meskipun tidak dibahas secara mendalam, integrasi data presensi dan lembur sangat berpengaruh terhadap akurasi perhitungan gaji, potongan, dan tunjangan.

2.1.3 Sistem HRIS (*Human Resource Information System*)

HRIS adalah sistem informasi yang digunakan untuk menyimpan, mengelola, dan memproses data sumber daya manusia secara terkomputerisasi. Pada studi kasus SISTAR, dinyatakan bahwa HRIS berfungsi mendukung pengambilan keputusan HR melalui penyediaan data yang cepat, lengkap, dan akurat (IJAEB, 2023).

> *"Human Resource Information System ... includes employee data, processing data, procedures, workflow, human resources and information technology to create fast, complete and accurate information to support personnel management."* (IJAEB, 2023, p. 38)

Digitalisasi memungkinkan proses HR menjadi lebih efisien, *data-driven*, serta meningkatkan pengalaman kerja karyawan (Kabul, 2024).Menurut jurnal 2025 tentang SISTAR:

a. Meningkatkan efisiensi administrasi HR.
b. Menyediakan data akurat dan *real-time*.
c. Mendukung proses pengambilan keputusan berbasis data.
d. Mengurangi kesalahan pencatatan manual.
e. Mempercepat proses rekap dan *monitoring* kinerja karyawan.

2.1.4 Konsep Presensi Elektronik

Presensi elektronik merupakan sistem pencatatan kehadiran yang dilakukan secara digital dan terintegrasi. Menurut IJAEB (2023), presensi elektronik telah menjadi bagian penting dari trans*form*asi digital dalam manajemen kehadiran. Salah satu contoh implementasi dapat dilihat pada Kementerian Agama RI yang melakukan perubahan dari sistem presensi berbasis *fingerprint* menjadi sistem presensi *online*. Sistem ini mencakup berbagai fitur pengendalian kehadiran karyawan, seperti pencatatan waktu datang dan pulang, pencatatan ketidakhadiran, pengajuan keluhan, hingga pencatatan tunjangan, yang semuanya dirangkum dalam satu sistem presensi daring (IJAEB, 2023, p. 36–37).

Dalam praktiknya, presensi elektronik dapat berbasis lokasi melalui GPS atau *geofencing* untuk memvalidasi posisi pengguna saat melakukan *check-in* maupun *check-out*. Validasi posisi ini sering kali memanfaatkan metode RESTful dengan keamanan token dan algoritma tertentu seperti Haversine untuk menghitung radius jarak yang diizinkan (Purnama, Kamal, dan Yadila, 2023). Selain itu, sistem juga dapat berbasis jaringan dengan memanfaatkan WiFi atau alamat IP, sehingga pengguna hanya dapat melakukan presensi ketika berada dalam jaringan kantor. Untuk meningkatkan keamanan dan mencegah kecurangan, presensi elektronik sering dilengkapi dengan autentikasi *visual* atau *QR Code*. Sementara itu, *input* manual biasanya hanya digunakan secara terbatas, misalnya untuk pengajuan izin atau koreksi presensi.

IJAEB (2023) menekankan bahwa validasi kehadiran merupakan aspek krusial dalam sistem presensi elektronik. Validasi ini dilakukan melalui pencatatan waktu otomatis, verifikasi lokasi, autentikasi identitas, serta pencatatan jejak audit (*audit trail*). Dengan adanya sistem ini, diharapkan efektivitas kerja dan manajemen kehadiran dapat meningkat (IJAEB, 2023, p. 37).

Secara umum, presensi elektronik mencakup berbagai fungsi penting, seperti pelacakan jam datang dan pulang, manajemen shift kerja, pengaturan toleransi keterlambatan, pencatatan izin dan ketidakhadiran, perhitungan lembur, serta perekapan otomatis yang terintegrasi dengan sistem penggajian. Integrasi ini menjadikan presensi elektronik tidak hanya sebagai alat pencatat kehadiran, tetapi juga sebagai bagian dari sistem manajemen sumber daya manusia yang lebih komprehensif.

2.2 **Sistem** Informasi

Menurut Laudon & Laudon (2022), *"an information system can be defined technically as a set of interrelated components that collect (or retrieve), process, store, and distribute information to support decision making and control in an organization."* Selain mendukung proses pengambilan keputusan, sistem informasi juga berperan dalam aktivitas koordinasi, pengendalian, analisis permasalahan,

*visual*isasi kondisi kompleks, serta mendukung organisasi dalam menciptakan produk atau layanan baru.

Sistem informasi memuat informasi terkait orang, tempat, dan objek yang relevan bagi organisasi. Informasi tersebut berasal dari data yang telah diolah sehingga bermakna, sedangkan data merupakan fakta mentah yang belum diproses dan belum memiliki arti (Laudon & Laudon, 2022). Dengan demikian, sistem informasi dapat dipahami sebagai kombinasi komponen-komponen yang bekerja secara terintegrasi untuk mengelola data menjadi informasi bernilai bagi proses bisnis, operasi, dan pengambilan keputusan.

2.2.1 Komponen Sistem Informasi

Komponen utama sistem informasi meliputi:

a. **Perangkat Keras (*Hardware*)** Perangkat fisik seperti komputer, *server*, perangkat jaringan, serta perangkat *input* dan *output*.
b. **Perangkat Lunak (Software)** Aplikasi atau program yang menjalankan fungsi sistem dan mengolah data.
c. **Basis Data (*Database*)** Kumpulan data terstruktur yang dapat diakses untuk mendukung proses sistem informasi.
d. **Sumber Daya Manusia (*People*)** Pengguna akhir dan tenaga ahli, seperti analis sistem, *developer*, administrator, serta teknisi.
e. **Prosedur (*Procedures*)** Aturan, tata *cara*, dan metode yang digunakan dalam pengoperasian sistem.
f. **Jaringan Komunikasi (*Network*)** Infrastruktur yang memungkinkan pertukaran data antar perangkat maupun antar sistem.

Komponen-komponen tersebut harus berfungsi secara sinergis agar sistem informasi dapat beroperasi secara optimal.

2.2.2 Konsep *Input–Process–Output* (IPO)

Model *Input–Process–Output* (IPO) menggambarkan alur dasar dalam pengolahan data:

a. *Input* — Data atau instruksi yang dimasukkan *ke* sistem.

b. *Process* — Pengolahan data melalui perhitungan, penyaringan, pengurutan, atau trans*form*asi.
c. *Output* — Informasi yang dihasilkan dan digunakan untuk pengambilan keputusan atau penyusunan laporan.

Konsep IPO membantu memvisualisasikan mekanisme kerja sistem informasi secara sistematis dan mudah dipahami.

2.2.3 Arsitektur *Client–Server* dan *Web Application*

Dalam pengembangan sistem informasi modern, pemilihan arsitektur menjadi aspek yang sangat menentukan kinerja dan skalabilitas aplikasi. Salah satu arsitektur yang paling banyak digunakan adalah *client–server*, di mana proses kerja dibagi antara perangkat pengguna dan *server* pusat. Seiring perkembangan teknologi internet, arsitektur ini kemudian berkembang menjadi web application architecture yang memungkinkan aplikasi dijalankan melalui *browser* dengan dukungan protokol *HTTP/HTTPS*. Kedua arsitektur ini menjadi fondasi utama bagi sistem informasi berbasis digital karena mampu mendukung integrasi layanan eksternal, autentikasi pengguna, serta komunikasi data secara *real-time*.

a. Arsitektur *Client–Server*

Arsitektur ini membagi proses kerja antara dua entitas:

a. ***Client***, yang mengirim permintaan (*request*), dan
b. ***Server***, yang memproses permintaan serta mengembalikan respons.

Model ini memiliki keunggulan dalam hal pembagian beban kerja, keamanan, dan skalabilitas sistem.

b. Arsitektur *Web Application*

Aplikasi web berjalan melalui *browser* menggunakan protokol *HTTP/HTTPS* dan umumnya memiliki tiga komponen utama:

a. ***Front-end* (*Client*-side)** — antarmuka pengguna yang ditampilkan melalui *browser*.
b. ***Back-end* (*Server*-side)** — logic aplikasi, pemrosesan data, dan penyediaan *API*.
c. ***Database*** — penyimpanan data yang dikelola dan diakses oleh *server*.

Aplikasi web modern mengadopsi model *client–server* yang mendukung integrasi layanan eksternal, autentikasi, dan komunikasi *real-time*.

2.3 Landasan Teori Teknologi *Backend* yang Digunakan

2.3.1 *Node JS*

Dalam pengembangan sistem *backend* modern, *Node.js* menjadi salah satu teknologi yang banyak digunakan karena per*forma* tinggi dan fleksibilitasnya. *Node.js* memungkinkan penggunaan bahasa *JavaScript* tidak hanya di sisi klien, tetapi juga di sisi *server*, sehingga konsistensi bahasa dapat terjaga.

2.3.1.1 Pengertian *Node JS*

Node.js adalah *open-source* dan *cross*-platform *JavaScript runtime environment* yang memungkinkan *JavaScript* dijalankan di luar *browser*. *Node.js* menggunakan mesin V8 milik *Google Chrome* yang mengeksekusi kode *JavaScript* secara cepat dan efisien. Dokumentasi resmi menyatakan bahwa:

"Node.js runs the V8 JavaScript engine… outside of the browser. This allows Node.js to be very performant." (*Node.js Documentation*, 2024)

Node.js juga populer karena memungkinkan *developer frontend* menggunakan bahasa yang sama untuk menulis *server-side code*.

"…frontend developers that write JavaScript for the browser are now able to write the server-side code…" (*Node.js Documentation*, 2024)

Dengan demikian, *Node.js* menjadi fondasi utama dalam pengembangan *backend* modern yang membutuhkan per*forma* tinggi, fleksibilitas, serta konsistensi bahasa pemrograman.

2.3.1.2 Arsitektur *Event-Driven* dan *Non-Blocking* I/O

Node.js berjalan dalam satu proses *single-thread* dan menggunakan arsitektur *event-driven* dengan mekanisme *non-blocking asynchronous I/O*. Pendekatan ini membuat *Node.js* mampu menangani banyak koneksi secara bersamaan tanpa membuat banyak *thread*.

Dokumentasi resmi menjelaskan:

"A Node.js app runs in a single process… Node.js provides asynchronous I/O primitives that prevent blocking." (*Node.js Documentation*, 2024)

Saat melakukan operasi seperti *query database*, *Node.js* tidak menunggu operasi selesai:

"…instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back." (*Node.js Documentation*, 2024)

Pendekatan ini membuat *Node.js* mampu menangani ribuan koneksi secara efisien:

"This allows Node.js to handle thousands of concurrent connections with a single server…" (*Node.js Documentation*, 2024)

2.3.1.3 Kelebihan *Node.js* sebagai *Backend*

Kelebihan *Node.js* meliputi:

a. Per*form*a tinggi karena mesin V8 yang cepat.
b. Skalabilitas tinggi melalui *event-driven* dan *non-blocking* I/O.
c. Efisiensi pengembangan (satu bahasa: *JavaScript*).
d. Ekosistem *library* yang besar melalui npm.
e. Dukungan standar *ECMAScript* terbaru.

Kelebihan ini menjadikan *Node.js* ideal untuk aplikasi *backend* seperti HRIS, *RESTful API*, atau sistem dengan banyak permintaan simultan.

2.3.2 *Express* JS

Untuk mempermudah pengembangan aplikasi berbasis *Node.js*, digunakan *framework Express.js*. . *Framework* ini menyediakan fitur *routing* dan *middleware* yang menjadikan pengembangan *RESTful API* lebih cepat dan terstruktur

2.3.2.1 Pengertian *Express* JS

Express.js adalah *framework* web minimalis dan fleksibel yang berjalan di atas *Node.js*. *Express* menyediakan fitur *routing*, *middleware*, dan manajemen objek *HTTP request*–response untuk membangun *RESTful API* secara cepat. *Framework* ini bersifat *unopinionated*, memberi kebebasan kepada pengembang untuk menentukan *struktur* aplikasi.

Menurut GeeksforGeeks (2025), *Express* mendukung integrasi *middleware* untuk autentikasi, *session*, keamanan, *parsing* data, *logging*, dan lain-lain.

2.3.2.2 *Routing* pada *Express* JS

Routing adalah mekanisme untuk menentukan bagaimana aplikasi merespons permintaan *HTTP* seperti *GET*, *POST*, *PUT*, dan *DELETE*.

Contoh dasar:

```
app.get('/users', (req, res) => {
  res.send('Menampilkan data pengguna');
});
```

Express juga mendukung modularisasi dengan express.Router() untuk membuat *struktur* rute yang lebih terorganisir (GeeksforGeeks, 2025).

2.3.2.3 *Middleware*

Middleware adalah fungsi yang memiliki akses *ke* req, res, dan next(). *Middleware* digunakan untuk:

1) Autentikasi dan validasi
2) *Logging* aktivitas *request*
3) *Parsing body request*
4) *Error* handling
5) Pengaturan *session* atau *cookie*

Contoh:

```
app.use((req, res, next) => {
  console.log('Request diterima:', req.method, req.url);
  next();
});
```

(GeeksforGeeks, 2025)

2.3.3 *RESTful API* dalam *Express* JS

Salah satu implementasi utama dari *Express* JS adalah *RESTful API*. *RESTful API* telah menjadi *stand*ar komunikasi antar sistem dan gaya arsitektur berbasis *HTTP* untuk operasi CRUD. *Express.js* sangat umum digunakan untuk membuat *RESTful API* karena *routing* yang fleksibel dan mekanisme pengelolaan *HTTP* yang efisien.

```
router.get('/employee-presences',authenticateToken,
employeePresence Controller.getAllEmployeePresences);

router.get('/employee-presences/:id',authenticateToken,
employeePresence Controller.getEmployeePresenceById);

router.post('/employee-presences',authenticateToken,
employeePresence Controller.createEmployeePresence);

router.put('/employee-presences/:id',authenticateToken,
employeePresence Controller.updateEmployeePresence);

router.delete('/employee-presences/:id',authenticateToken,
employeePresence Controller.deleteEmployeePresence);
```

(GeeksforGeeks, 2025)

2.3.4 *PostgreSQL*

Dalam sistem *backend*, pengelolaan data menjadi aspek krusial. *PostgreSQL* dipilih sebagai sistem manajemen basis data karena keandalannya dalam menjaga integritas data serta dukungan fitur *relasional* yang lengkap.

2.3.4.1 Pengertian *PostgreSQL*

PostgreSQL merupakan sistem manajemen basis data (*Database Management System*/DBMS) berjenis *object-relational* yang bersifat *open source* serta mampu menangani pengelolaan data dalam skala kecil hingga besar. *PostgreSQL* dikembangkan dari proyek *POSTGRES* di University of California, Berkeley sejak tahun 1986, dan hingga sekarang terus mengalami pengembangan aktif selama hampir empat dekade.

PostgreSQL dikenal memiliki arsitektur yang stabil, keandalan tinggi dalam menjaga integritas data, dan kemampuan skalabilitas yang baik. DBMS ini mendukung berbagai fitur seperti transaksi ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*), replikasi data, indexing tingkat lanjut, serta ekstensibilitas melalui tipe data maupun fungsi kustom. *PostgreSQL* dapat dijalankan pada berbagai sistem operasi modern dan digunakan secara luas di lingkungan industri, akademik, maupun riset. Pemilihan DBMS yang stabil sangat krusial karena basis data merupakan komponen inti sistem informasi yang berfungsi mengorganisasikan data secara terstruktur agar dapat diakses secara efisien (Laudon & Laudon, 2022).

2.3.4.2 Konsep *Database Relasional*

PostgreSQL merupakan *Relational Database Management System* (RDBMS), yaitu sistem basis data yang menyimpan data dalam bentuk tabel dan menghubungkan satu tabel dengan tabel lainnya menggunakan *key* tertentu. Konsep *relasional* memastikan data tersimpan secara terstruktur sehingga mudah diakses, diproses, dan dikelola menggunakan SQL (*Structured Query Language*).

Pada *database relasional*, data diorganisasikan dalam:

a. **Tabel**, terdiri dari baris dan kolom
b. **Relasi**, yaitu hubungan antar tabel berdasarkan atribut terkait
c. ***Schema***, yaitu *struktur* logis dari basis data
d. ***Query***, yaitu perintah SQL untuk mengakses atau memanipulasi data

Model *relasional* membantu mengurangi redundansi data serta menjaga konsistensi informasi di dalam sistem. Laudon dan Laudon (2022) mendefinisikan sistem manajemen basis data relasional sebagai metode pengorganisasian data ke dalam tabel dua dimensi yang terdiri dari kolom dan baris yang saling berhubungan.

2.3.4.3 *Primary Key*, *Foreign Key*, *Constraint*s, dan *Transaction*

Beberapa konsep penting dalam *PostgreSQL* dan RDBMS lainnya antara lain:

a. ***Primary Key* (PK)**

   Merupakan atribut unik dalam tabel yang digunakan untuk mengidentifikasi setiap baris data secara spesifik. Setiap tabel umumnya memiliki satu PK dan nilainya tidak boleh duplikat maupun bernilai NULL.

b. ***Foreign Key* (FK)**

Merupakan atribut yang berfungsi untuk menghubungkan satu tabel dengan tabel lainnya berdasarkan referensi nilai dari *Primary Key*. FK digunakan untuk menjaga integritas hubungan antar tabel.

c. ***Constraint***

Merupakan aturan yang diterapkan pada tabel untuk memastikan kualitas dan konsistensi data. Jenis *constraint* antara lain:

1) *NOT NULL*
2) *UNIQUE*
3) *CHECK*
4) *PRIMARY KEY*
5) *FOREIGN KEY*
6) *EXCLUSION*

Constraint membantu mencegah kesalahan *input* dan kerusakan data.

d. ***Transaction***

Transaksi adalah serangkaian operasi *database* yang diperlakukan sebagai satu kesatuan proses. *PostgreSQL* mendukung **ACID *Transaction*** sehingga setiap operasi data dapat dipastikan aman dan konsisten. Operasi transaksi umum:

1) *BEGIN*
2) *COMMIT*
3) *ROLLBACK*

PostgreSQL juga mendukung ***savepoint*** untuk transaksi bertingkat.

2.3.4.4 Normalisasi Data

Normalisasi merupakan proses penyusunan *struktur* tabel agar lebih efisien, mengurangi redundansi, dan meningkatkan integritas data. Normalisasi dilakukan dengan memecah data *ke* dalam beberapa tabel yang saling berhubungan.

Tingkat normalisasi yang umum digunakan meliputi:

a. **1NF (*First* Normal *Form*)** – setiap kolom memiliki satu nilai (atomic).
b. **2NF (*Second* Normal *Form*)** – setiap atribut non-*key* bergantung penuh pada *primary key*.
c. **3NF (*Third* Normal *Form*)** – tidak ada dependensi antar atribut non-*key*.

2.3.5 *Docker*

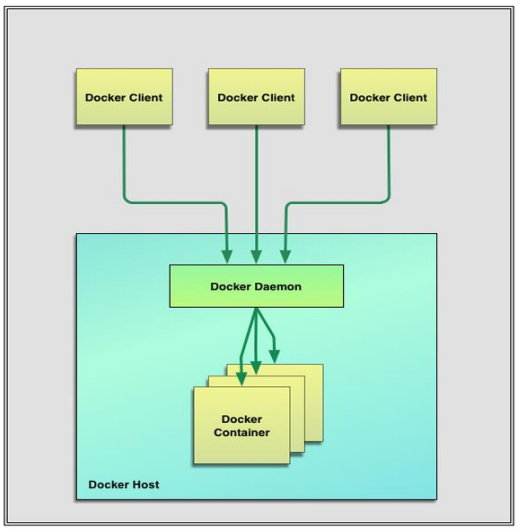
Agar aplikasi dapat berjalan konsisten di berbagai lingkungan, digunakan teknologi *container* melalui *Docker*. *Docker* memungkinkan proses *deployment* menjadi lebih cepat dan efisien.

2.3.5.1 Pengertian *Docker*

Docker adalah platform *open-source* yang digunakan untuk mengotomatisasi proses pembangunan, pengemasan, dan *deployment* aplikasi menggunakan teknologi *container*. *Docker* dikembangkan oleh *Docker*, Inc. dan dirilis di bawah lisensi Apache 2.0.

Dengan *Docker*, aplikasi dapat dijalankan dalam lingkungan yang terisolasi dan konsisten, tanpa bergantung pada konfigurasi sistem operasi *host*. Hal ini menjadikan *Docker* sangat efektif untuk menjaga keseragaman lingkungan antara development, *testing*, hingga production.

Docker bekerja di atas teknologi virtualisasi berbasis *container*, yang lebih ringan dibandingkan virtual *machine* karena tidak memerlukan sistem operasi lengkap di setiap *instance*. Pendekatan ini meningkatkan kecepatan *deployment* sekaligus efisiensi penggunaan sumber daya. Docker memungkinkan aplikasi dikemas dalam sebuah *image* yang ringan dan siap dijalankan di mana saja (dskcode, no date a).



Gambar 2. 1 Diagram Arsitektur Docker (Sumber: James Turnbull, 2017)

Gambar 2.1 menunjukkan bahwa *Docker* berjalan di dalam sebuah *Docker Host* yang di dalamnya terdapat *Docker Daemon* sebagai komponen utama yang bertugas mengelola *container*, image, net*work*, dan volume. *Docker Daemon* berkomunikasi langsung dengan *Docker Client*, yaitu antarmuka yang digunakan oleh pengguna untuk memberikan perintah seperti pembuatan, menjalankan, menghentikan, dan mengelola *container*. *Docker Client* berada di luar *Docker Host* dan berinteraksi dengan *Docker Daemon* melalui *Docker API*, baik secara lokal maupun *remote*. *Container Docker* yang dijalankan di dalam *Docker Host* berisi aplikasi beserta seluruh dependensinya, sehingga aplikasi dapat berjalan secara terisolasi dan konsisten tanpa bergantung pada lingkungan sistem operasi *host*.

2.3.5.2 Konsep *Container* pada *Docker*

Container adalah unit eksekusi yang berisi aplikasi beserta seluruh dependensi yang dibutuhkan untuk berjalan dengan konsisten. *Docker* memanfaatkan fitur kernel Linux seperti:

a. ***Namespaces*** → memberikan isolasi proses, jaringan, dan sistem file
b. ***Control* Gro*up*s (cgro*up*s)** → mengatur alokasi sumber daya seperti CPU, memori, dan I/O

Container dapat dibuat dan dijalankan dengan cepat karena menggunakan mekanisme **copy-on-write**, sehingga hanya perubahan konfigurasi yang disimpan. Konsep ini membuat *container* jauh lebih efisien dibanding VM.

2.3.5.3 Komponen Utama *Docker*

a. *Docker* Engine (*Client* & *Server*)
   Docker menggunakan arsitektur *client-server*.
   - ***Docker Client*** menerima perintah dari pengguna.
   - ***Docker Daemon* (*docker*d)** menjalankan, membuat, dan mengelola *container* serta image.

   Keduanya berkomunikasi melalui *REST API*.
b. *Docker* Image
   Image merupakan blueprint berisi aplikasi dan dependensinya. Image bersifat read-only dan digunakan untuk membuat *container*.

c. *Docker Container*

   Container adalah *instance* berjalan dari image. *Container* dapat di-*start*, *stop*, *restart*, dan *remove* tanpa memengaruhi image asli.

d. *Docker* Registry

e. Registry adalah repositori penyimpanan image.

   Contoh:

   a. *Docker* Hub (publik)

   b. *Git*Hub *Container* Registry

   c. Registry privat perusahaan

2.3.5.4 *Docker Compose*

Docker Compose adalah tool untuk menjalankan beberapa *container* secara bersamaan menggunakan satu file konfigurasi ***docker-compose*.yml**. *Compose* sangat berguna untuk project yang memiliki beberapa service, seperti:

d. *Backend* (*Express.js*)

e. *Frontend*

f. *Database* (*PostgreSQL*)

Dengan *Compose*, seluruh layanan dapat dijalankan dengan satu perintah:

docker compose up -d

2.3.5.5 Manfaat *Docker* dalam Pengembangan

Penggunaan *Docker* memberikan berbagai keuntungan, terutama pada tim development:

a. Konsistensi environment antara *developer* dan *server* produksi

b. Mempercepat proses *deployment*

c. Mengurangi masalah konfigurasi antar mesin

d. Portabilitas tinggi karena *container* dapat dijalankan di berbagai platform

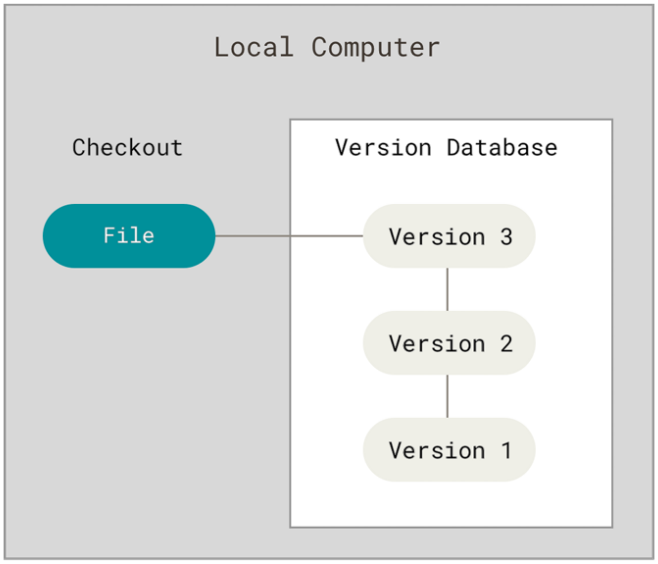
e. Mendukung arsitektur microservices

f. Lebih efisien dibanding VM

g. Mendukung pipeline DevOps dan CI/CD

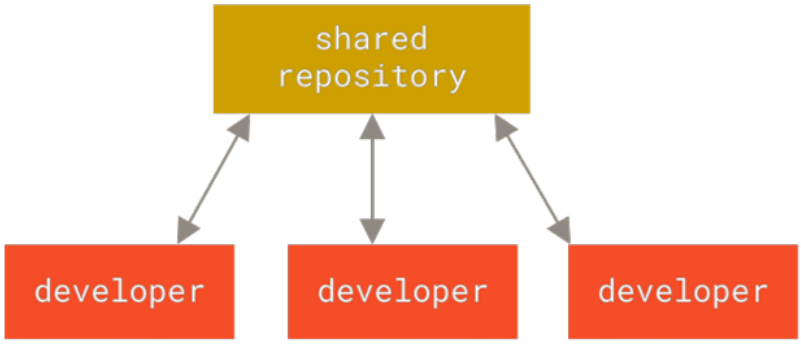
2.3.6 *Git & Version Control*

Dalam pengembangan perangkat lunak secara tim, pengelolaan versi kode sangat penting. *Git* sebagai sistem version control terdistribusi digunakan untuk mencatat perubahan, mendukung kolaborasi, serta menjaga riwayat pengembangan proyek.



Gambar 2. 2 Local Version Control Diagram (Sumber: Chacon & Straub, no date)

Gambar 2.2 menggambarkan proses pengelolaan versi file yang dilakukan secara lokal pada satu komputer pengembang tanpa melibatkan *server* terpusat. Pada *diagram* tersebut, *local computer* berperan sebagai lingkungan kerja utama tempat pengembang melakukan pengembangan dan perubahan terhadap file proyek. Proses *checkout file* dilakukan dengan mengambil versi tertentu dari *version database ke* direktori kerja lokal untuk diedit atau diperbarui. Setiap perubahan yang disimpan akan dicatat sebagai versi baru di dalam *version database*, yang ditunjukkan dengan penomoran versi secara berurutan seperti versi 1, versi 2, hingga versi 3. Dengan mekanisme ini, pengembang dapat melacak perubahan, kembali *ke* versi sebelumnya apabila terjadi kesalahan, serta menjaga konsistensi file selama proses pengembangan berlangsung.



Gambar 2. 3 Controller Version Control Diagram (Sumber: Chacon & Straub, no date)

Gambar 2.3 menggambarkan sistem pengelolaan versi yang menggunakan satu *shared repository* sebagai pusat penyimpanan kode sumber. Pada arsitektur ini, seluruh pengembang bekerja dengan terhubung *ke repository* terpusat yang sama. Setiap *developer* melakukan proses *checkout* untuk mengambil versi terbaru kode dari *shared repository ke* komputer lokal masing-masing, kemudian melakukan perubahan sesuai dengan tugas yang diberikan. Setelah perubahan selesai, *developer* melakukan *commit* kembali *ke repository* pusat sehingga versi kode diperbarui dan dapat diakses oleh *developer* lainnya. Dengan adanya *repository* terpusat, proses kolaborasi antar tiga *developer* dapat terkoordinasi dengan baik, konflik perubahan dapat dikelola, dan konsistensi versi kode dapat tetap terjaga selama proses pengembangan sistem.

2.3.6.1 Pengertian *Version Control*

Version Control System (VCS) adalah sistem yang digunakan untuk mencatat perubahan pada file atau sekumpulan file sepanjang waktu. Melalui VCS, pengguna dapat melacak sejarah perubahan, kembali *ke* versi sebelumnya, membandingkan versi, serta mengetahui siapa dan kapan perubahan dilakukan.

Sistem ini sangat penting dalam pengembangan perangkat lunak karena mendukung kolaborasi antar *developer* dan meminimalkan risiko kehilangan data. Selain itu, VCS membantu proses pemulihan ketika terjadi kesalahan atau kerusakan pada file proyek (Chacon & Straub, *Pro Git*).

2.3.6.2 Distributed *Version Control System* (DVCS)

Pada Distributed *Version Control System* (DVCS) seperti *Git*, setiap pengguna menyimpan copy lengkap dari *repository* — bukan hanya versi terbaru, tetapi

seluruh riwayatnya.

Dengan demikian, setiap clone *repository* dapat menjadi *backup* penuh apabila *server* utama mengalami gangguan.

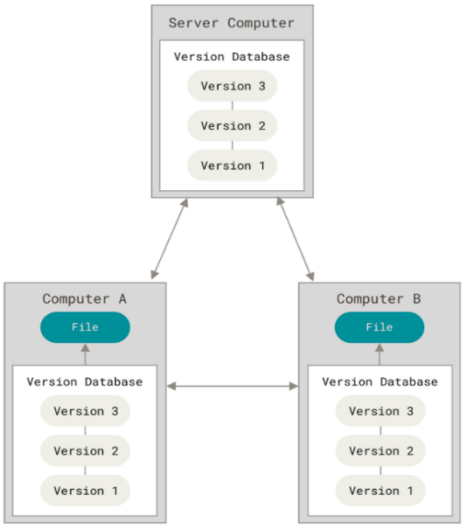
DVCS juga mendukung penggunaan banyak *remote*, sehingga kolaborasi dapat dilakukan secara fleksibel dan paralel di berbagai tim dalam satu proyek.

2.3.6.3 Pengertian *Git*

Git adalah sistem version control terdistribusi yang dikembangkan oleh Linus Torvalds pada tahun 2005 untuk mendukung pengembangan kernel Linux. *Git* dirancang dengan fokus pada:

a. kecepatan,
b. kesederhanaan desain,
c. dukungan pengembangan non-linear (ribuan branch paralel),
d. efisiensi penyimpanan,
e. serta ketahanan terhadap kerusakan.

Git menyimpan data sebagai *snapshot* lengkap dari file pada setiap *commit*, bukan sekadar perbedaan baris. Pendekatan ini membuat *Git* lebih cepat dan andal untuk proyek modern.



Gambar 2. 4 Distributed Version Control Diagram ((Sumber: Chacon & Straub, no date)

Gambar 2.4 menggambarkan sistem pengelolaan versi di mana setiap pengembang memiliki salinan *repository* secara lengkap pada komputer masing-masing. Pada *diagram* tersebut, *server computer* berfungsi sebagai *repository* pusat yang digunakan untuk berbagi dan menyinkronkan perubahan, sedangkan *computer A* dan *computer B* merupakan lingkungan kerja para pengembang yang masing-masing menyimpan *repository* lokal beserta riwayat versinya. Setiap pengembang dapat melakukan *commit* perubahan secara mandiri di komputer lokal tanpa harus selalu terhubung *ke server*. Selanjutnya, perubahan yang telah dilakukan akan disinkronkan *ke server computer* melalui proses *push* dan dapat diambil oleh pengembang lain melalui proses *pull*, sehingga kolaborasi dapat berjalan secara fleksibel, efisien, dan tetap menjaga konsistensi versi kode sumber.

2.3.6.4 Konsep *Commit*, Branch, dan Merge

a. *Commit*, snapshot perubahan yang disimpan *ke database Git*. Memuat identitas pembuat dan pesan *commit*.
b. Branch, cabang pengembangan yang memungkinkan pembuatan fitur secara terpisah tanpa mengganggu kode utama
c. Merge, proses menggabungkan perubahan dari satu branch *ke* branch lain (misalnya feature → develop/main).

2.3.6.5 Tiga Status Utama dalam *Git*

Git memiliki tiga status file:

a. **Modified** → file telah diubah tetapi belum masuk staging.
b. **Staged** → file telah disiapkan untuk *commit*.
c. ***Commit*ted** → perubahan telah disimpan permanen di *repository*.

Alur kerja dasar:

a. Mengubah file pada *working tree*,
b. Menambahkan *ke* staging area,
c. Melakukan *commit ke repository* lokal.

2.3.6.6 *Work*flow Kolaborasi Tim Menggunakan *Git*

Dalam pengembangan perangkat lunak secara tim, *Git* Flow sering digunakan untuk memisahkan tahapan pengembangan:

Tabel 2. 1 Tabel Workflow Kolaborasi Tim Menggunakan Git

Branch	Fungsi
main/master	Berisi versi stabil siap produksi
develop	Area integrasi fitur sebelum rilis
feature/	Pengembangan fitur baru
release/	Persiapan final sebelum rilis
hotfix/	Perbaikan cepat pada versi produksi

(Sumber: Hasil Olahan sendiri, 2025)

Tabel 2.1 diatas menjelaskan alur kerja tim (ringkas) seperti berikut:

a. Membuat branch feature dari develop.
b. Mengembangkan dan melakukan *commit* bertahap.
c. Merge fitur kembali *ke* develop.
d. Membuat branch release untuk persiapan rilis.
e. Setelah stabil, merge *ke* main dan develop.
f. Jika ada bug kritis, gunakan branch hotfix.

2.4 Metodologi Pengembangan

Metodologi pengembangan merupakan kerangka kerja yang digunakan untuk mengelola proses pembuatan perangkat lunak secara sistematis. Dalam pengembangan sistem ini, metodologi yang digunakan harus mampu mendukung perubahan kebutuhan teknis secara cepat dan memfasilitasi kolaborasi antar pengembang. Pemilihan metodologi yang tepat sangat menentukan efisiensi manajemen proyek dan kualitas produk akhir yang dihasilkan.

2.4.1 Metode *Agile*

Metode *Agile* dipilih karena karakteristiknya yang adaptif terhadap dinamika pengembangan sistem informasi di lingkungan organisasi. Pendekatan ini

memungkinkan tim pengembangan untuk memberikan hasil secara inkremental dan terus melakukan perbaikan berdasarkan umpan balik dari pengguna.

2.4.1.1 Definisi

Agile adalah metodologi pengembangan perangkat lunak yang menekankan pada fleksibilitas, kolaborasi, dan pengiriman produk secara iteratif. *Agile* muncul sebagai respon terhadap keterbatasan metode tradisional (waterfall) yang cenderung kaku. Dengan *Agile*, proyek dibagi menjadi iterasi kecil (sprint atau iteration) sehingga tim dapat beradaptasi dengan cepat terhadap perubahan kebutuhan dan memberikan nilai secara berkelanjutan kepada pengguna. Pengembangan perangkat lunak *Agile* menekankan pada pengiriman *software* yang berfungsi dalam interval waktu yang singkat (Winter, 2014; y-t99, no date).

2.4.1.2 Prinsip *Agile*

Agile berlandaskan pada *Agile* Manifesto (2001) yang memuat empat nilai utama (Manifesto Pengembangan Perangkat Lunak Agile, no date) seperti berikut:

a. Individu dan interaksi lebih penting daripada proses dan alat.
b. Perangkat lunak yang berfungsi lebih penting daripada dokumentasi yang lengkap.
c. Kolaborasi dengan pelanggan lebih penting daripada negosiasi kontrak.
d. Merespons perubahan lebih penting daripada mengikuti rencana.

Selain itu, terdapat 12 prinsip *Agile* yang menjadi pedoman, di antaranya:

a. Kepuasan pelanggan melalui pengiriman perangkat lunak secara berkelanjutan.
b. Menyambut perubahan kebutuhan, bahkan di tahap akhir pengembangan.
c. Kolaborasi erat antara pengembang dan pemilik bisnis.
d. Tim yang termotivasi dan diberi kepercayaan.
e. Komunikasi tatap muka sebagai *cara* paling efektif.
f. Perangkat lunak yang berfungsi sebagai ukuran utama kemajuan.
g. Pengembangan berkelanjutan dengan tempo yang konsisten.
h. Kesederhanaan sebagai seni memaksimalkan pekerjaan yang tidak dilakukan.

2.4.1.3 Keuntungan *Agile*

Metode *Agile* memiliki sejumlah keunggulan dibandingkan metode tradisional:

a. **Fleksibilitas tinggi** dalam menghadapi perubahan kebutuhan sistem.
b. **Iteratif dan inkremental**, sehingga hasil dapat dievaluasi secara berkala.
c. **Kolaboratif**, melibatkan stakeholder secara aktif dalam setiap tahap.
d. **Cepat menghasilkan nilai**, karena setiap iterasi menghasilkan produk yang bisa digunakan.
e. **Meningkatkan kepuasan pengguna**, karena produk dikembangkan sesuai kebutuhan aktual.
f. **Mengurangi risiko kegagalan proyek**, karena masalah dapat diidentifikasi lebih awal.

Keterkaitan dengan Proyek PKL dalam konteks Pengembangan Modul *Backend* Presensi dan *Human Resource Management* Terintegrasi dengan Sistem Penggajian, penerapan *Agile* sangat relevan karena:

a. Modul dapat dikembangkan secara bertahap (misalnya presensi → HRM → penggajian).
b. Stakeholder (HR, finance, karyawan) dapat memberikan masukan di setiap iterasi.
c. Perubahan kebijakan perusahaan (misalnya aturan absensi atau sistem penggajian) dapat segera diakomodasi.

"*Agile* methodology is a project *management framework* that breaks projects down into several dynamic phases" (Asana, 2025).

2.4.2 *Scrum Framework*

Scrum merupakan salah satu kerangka kerja paling populer dalam metodologi *Agile* yang memfasilitasi pengembangan produk secara adaptif dan kolaboratif. Kerangka kerja ini tidak bersifat instruktif secara kaku, melainkan menyediakan batasan (*boundaries*) yang memungkinkan tim untuk terus belajar dan melakukan perbaikan proses secara mandiri guna mencapai tujuan produk yang berkualitas.

2.4.2.1 Definisi

Scrum adalah sebuah lightweight *framework* yang digunakan untuk mengelola dan mengembangkan produk kompleks secara adaptif. *Framework* ini berfungsi sebagai wadah untuk berbagai praktik dan teknik, dengan tujuan utama menghasilkan nilai secara berkelanjutan melalui iterasi yang disebut Sprint. *Scrum* berlandaskan pada tiga pilar empirisme: transparansi, inspeksi, dan adaptasi (Scrum Guide, no date).

2.4.2.2 Roles

Scrum Team terdiri dari tiga peran utama:

a. **Product Owner** Bertanggung jawab memaksimalkan nilai produk melalui pengelolaan *Product Backlog*. Product Owner memastikan backlog transparan, jelas, dan dipahami oleh semua pihak.
b. ***Scrum* Master** Bertanggung jawab memastikan *Scrum* dipahami dan dijalankan sesuai prinsip. *Scrum* Master berperan sebagai fasilitator, coach, dan servant leader bagi tim maupun organisasi.
c. ***Development* Team (*Developer*s)** Bertanggung jawab menghasilkan *Increment* yang memenuhi *Definition of Done*. Tim ini bersifat **cross-functional** dan **self-managing**, sehingga menentukan sendiri bagaimana pekerjaan dilakukan.

2.4.2.3 Artifacts

Scrum memiliki tiga artefak utama yang berfungsi menjaga transparansi dan fokus:

a. **Product Backlog** Daftar pekerjaan yang bersifat dinamis dan terus berkembang untuk mencapai *Product Goal*.
b. **Sprint Backlog** Rencana kerja tim selama satu Sprint, terdiri dari *Sprint Goal*, item backlog yang dipilih, serta rencana penyelesaiannya.
c. **Increment** Hasil kerja yang bernilai dan dapat digunakan, yang memenuhi *Definition of Done*. Setiap Sprint menghasilkan Increment yang menambah nilai produk.

2.4.2.4 Events

Scrum memiliki lima event utama yang berfungsi sebagai wadah inspeksi dan adaptasi:

a. **Sprint** – inti dari *Scrum*, berlangsung maksimal satu bulan, menghasilkan Increment.
b. **Sprint Planning** – perencanaan pekerjaan Sprint, menentukan *Sprint Goal* dan backlog yang dipilih.
c. **Daily *Scrum* (*Daily Stand Up*)** – pertemuan harian 15 menit untuk memantau progres dan menyesuaikan rencana.
d. **Sprint Review** – evaluasi hasil Sprint bersama stakeholder, sekaligus menyesuaikan backlog.
e. **Sprint Retrospective** – refleksi tim untuk meningkatkan kualitas dan efektivitas kerja.

BAB III

HASIL PELAKSANAAN MAGANG

Bab ini membahas hasil pelaksanaan kegiatan Praktik Kerja Lapangan yang dilakukan di PT Naramedic Mitra Utama Solution. Uraian pada bab ini mencakup gambaran umum perusahaan, deskripsi kegiatan magang, serta pembahasan hasil pengembangan sistem yang telah dilakukan. Selain itu, dijelaskan pula peran dan tanggung jawab dalam pengembangan *backend* sistem HRIS Naraworks, penerapan metodologi pengembangan perangkat lunak, alur kerja teknis, serta desain dan implementasi sistem yang dihasilkan. Pada bagian akhir bab ini disajikan hasil pengujian sistem, kendala yang dihadapi selama proses pengembangan, serta solusi yang diterapkan sebagai *up*aya penyempurnaan sistem.

3.1 Profil DUDI

PT Naramedic Mitra Utama Solution Indonesia merupakan perusahaan yang bergerak di bidang inovasi kesehatan dan teknologi medis. Perusahaan ini berdiri dengan tujuan menghadirkan solusi kesehatan yang modern, efektif, dan terjangkau bagi masyarakat Indonesia. Naramedic berfokus pada pengembangan produk-produk kesehatan berbasis teknologi serta layanan yang mendukung peningkatan kualitas hid*up*. Dalam menjalankan operasionalnya, perusahaan memiliki beberapa divisi utama, yaitu riset dan pengembangan (R&D), teknologi informasi, operasional, serta pemasaran. Divisi-divisi tersebut bekerja secara sinergis untuk menghasilkan produk dan layanan yang sesuai dengan kebutuhan pasar dan perkembangan teknologi.

Sebagai bagian dari dunia usaha dan dunia industri (DUDI), Naramedic berperan aktif dalam mendukung kegiatan pendidikan melalui program magang. Mahasiswa yang melaksanakan praktik kerja lapangan (PKL) di perusahaan ini mendapatkan kesempatan untuk terlibat langsung dalam proses pengembangan sistem, khususnya pada modul *backend* presensi dan *human resource management* yang terintegrasi dengan sistem penggajian. Lingkungan kerja yang inovatif dan kolaboratif

memberikan pengalaman berharga bagi peserta magang dalam memahami penerapan metodologi pengembangan perangkat lunak modern seperti *Agile* dan *Scrum*, sekaligus memperkenalkan praktik nyata dalam industri kesehatan berbasis teknologi.

3.2 Uraian Kegiatan Magang

Selama melaksanakan kegiatan magang di PT Naramedic Mitra Utama Solution Indonesia, penulis ditempatkan pada divisi Teknologi Informasi dengan fokus pada pengembangan sistem *backend*. Aktivitas magang dilakukan dengan mengikuti alur kerja perusahaan yang menerapkan metodologi *Agile* dan *Scrum*. Setiap hari dimulai dengan *daily stand-up* meeting untuk membahas progres pekerjaan, kendala yang dihadapi, serta rencana penyelesaian. Penulis terlibat dalam proses analisis kebutuhan sistem, perancangan modul, implementasi kode, integrasi dengan sistem *human resource management*, serta pengujian modul yang dikembangkan.

Selain kegiatan rutin, penulis juga berpartisipasi dalam sprint planning dan sprint review bersama tim pengembang. Hal ini memberikan pengalaman nyata dalam memahami siklus pengembangan perangkat lunak berbasis iterasi. Kegiatan magang tidak hanya berfokus pada aspek teknis, tetapi juga melibatkan komunikasi dengan stakeholder untuk memastikan sistem yang dikembangkan sesuai kebutuhan perusahaan.

Untuk memberikan gambaran lebih jelas, berikut adalah ringkasan kegiatan magang yang dilakukan penulis selama periode magang:

Tabel 3. 1 Tabel Kegiatan Mingguan

Minggu	Kegiatan Utama	*Output* Deliverable
Minggu *ke* - 1	Orientasi lingkungan kerja, pengenalan *struktur* organisasi, alur bisnis perusahaan, serta *tools*	Pemahaman alur kerja perusahaan dan dokumentasi awal lingkungan

	pengembangan yang digunakan	pengembangan
Minggu *ke* - 2	Pengenalan sistem *Human Resource Management* (HRM) yang sedang dikembangkan, analisis kebutuhan modul presensi dan lembur	Dokumen analisis kebutuhan awal modul presensi dan lembur
Minggu *ke* - 3	Perancangan *database* awal untuk modul presensi dan lembur, termasuk relasi tabel dan *constraint*	*Skema database* dan ERD modul presensi & lembur
Minggu *ke* - 4	Implementasi *backend* dasar modul presensi karyawan menggunakan *Node.js* dan *PostgreSQL*	Endpoint *API* presensi dasar (create & read data presensi)
Minggu *ke* - 5	Pengembangan fitur validasi presensi, pengolahan waktu masuk dan keluar, serta penyesuaian timezone	Logika validasi presensi dan pencatatan waktu kerja
Minggu *ke* - 6	Implementasi modul lembur karyawan dan integrasinya dengan data presensi	Endpoint *API* lembur terintegrasi dengan presensi
Minggu *ke* - 7	Pengujian modul presensi dan lembur (unit *testing* & manual *testing*), serta	Modul presensi dan lembur yang stabil

	perbaikan bug	
Minggu *ke* - 8	Sprint review dan evaluasi modul yang telah dikembangkan bersama tim dan stakeholder	Feedback pengembangan dan catatan perbaikan sistem
Minggu *ke* - 9	Pengembangan modul approval lembur oleh atasan dan HR	Endpoint approval lembur dan alur status lembur
Minggu *ke* - 10	Integrasi modul presensi dan lembur dengan sistem *payroll*	Sinkronisasi data presensi & lembur *ke* perhitungan gaji
Minggu *ke* - 11	Implementasi logika perhitungan jam kerja, keterlambatan, dan lembur	Fungsi perhitungan jam kerja dan lembur otomatis
Minggu *ke* - 12	Penyusunan dan perbaikan dokumentasi *API* (*API* contract & *endpoint* documentation)	Dokumentasi *endpoint API backend*
Minggu *ke* - 13	Konfigurasi environment *backend* menggunakan *Docker* dan *Docker Compose*	*Backend* berjalan dalam *container Docker*
Minggu *ke* – 14	Debugging error konfigurasi environment, *database* connection, dan variabel .env	Sistem *backend* berjalan stabil di environment *container*
Minggu *ke* - 15	Uji integrasi menyeluruh	Sistem *backend*

	modul presensi, lembur, dan *payroll*	terintegrasi secara end-to-end
Minggu *ke* – 16	Finalisasi proyek magang, evaluasi akhir bersama pembimbing industri, dan penyusunan laporan	Sistem *backend* siap digunakan dan laporan magang

(Sumber: Hasil olahan sendiri, 2025)

3.3 Pembahasan Hasil Magang

Selama magang di PT Naramedic Mitra Utama Solution, penulis berfokus pada pengembangan modul *backend* Naraworks HRIS yang mengelola presensi, HRM, dan integrasi penggajian menggunakan *Express.js* dan *PostgreSQL*. Pembahasan ini mencakup peran, metodologi, dan alur kerja yang diterapkan untuk memastikan sistem berjalan efisien dan terintegrasi.

3.3.1 Peran dan Tanggung Jawab

Penulis bertanggung jawab atas pengembangan *server-side* menggunakan *Node.js* dan *Express.js*, termasuk logika bisnis untuk presensi harian, lembur, dan kunjungan kerja. Fokus ini memastikan *backend* menangani pemrosesan data secara aman dan skalabel tanpa bergantung pada *frontend*.

a. Fokus pengerjaan pada sisi *backend* (*server-side*)

Penulis bertanggung jawab atas pengembangan *server-side* menggunakan *Node.js* dan *Express.js*, termasuk logika bisnis untuk presensi harian, lembur, dan kunjungan kerja. Fokus ini memastikan *backend* menangani pemrosesan data secara aman dan skalabel tanpa bergantung pada *frontend*.

b. Tanggung jawab dalam pembuatan *API* (*Application* Programming Interface)

Penulis merancang dan mengimplementasikan *RESTful API endpoint* seperti /employee-presences untuk operasi CRUD presensi, lengkap dengan *middleware* autentikasi *JWT* dan validasi *input*. *API* ini mendukung integrasi dengan modul *payroll* untuk perhitungan gaji otomatis.

c. Kolaborasi dengan tim *Frontend* (Konsumsi *API*) dan tim *Database*

Kolaborasi dilakukan melalui *Git*Hub untuk *code* review dan sinkronisasi *API* dengan *frontend* web/*mobile*, serta diskusi *skema database PostgreSQL* dengan tim DB untuk menjaga integritas data seperti relasi one-to-many antara tabel user dan attendance

3.3.2 Penerapan Metodologi Pengembangan

Dalam melaksanakan proyek pengembangan sistem *backend* HRIS Naraworks, PT Naramedic mengadopsi metodologi pengembangan perangkat lunak yang sistematis untuk menjamin fleksibilitas dan kecepatan pengiriman produk. Kerangka kerja yang dipilih bertujuan untuk memfasilitasi komunikasi yang intensif antara pengembang dan pemangku kepentingan, serta memungkinkan penyesuaian fitur secara dinamis berdasarkan umpan balik berkelanjutan. Implementasi metodologi ini mencakup siklus kerja rutin dan penggunaan perangkat pendukung manajemen proyek sebagai berikut:

a. Implementasi *Agile Scrum* di PT Naramedic

PT Naramedic menerapkan *Agile Scrum* dengan sprint 2 minggu untuk iterasi cepat, memungkinkan adaptasi terhadap perubahan kebutuhan HR seperti aturan lembur baru. Pendekatan ini meningkatkan kolaborasi tim dan pengiriman fitur bertahap.

b. Keterlibatan dalam Daily Stand-Up, Sprint Planning, dan Sprint Review

Penulis aktif dalam *daily stand-up* 15 menit untuk lapor progres dan blokir, sprint planning untuk prioritas backlog, serta sprint review untuk demo *API* dan feedback stakeholder. Ini memastikan alignment dengan tujuan proyek HRIS.

c. Penggunaan Project *Management Tools*

Tools seperti *Git*Hub untuk version control, *Postman* untuk *API testing*, dan *Docker Compose* untuk environment konsisten digunakan sepanjang siklus pengembangan. Alat ini mendukung kolaborasi *hybrid remote-onsite* dan CI/CD pipeline.

3.3.3 Alur Kerja Pengembangan *Backend*

Proses pengembangan *backend* untuk sistem HRIS Naraworks dilakukan melalui tahapan sistematis yang mengacu pada siklus hidup pengembangan perangkat lunak (*Software Development Life Cycle*). Setiap tahapan dirancang untuk memastikan bahwa kebutuhan bisnis dari departemen SDM dapat diterjemahkan ke dalam arsitektur teknis yang stabil dan aman. Seluruh alur kerja teknis ini dijalankan secara iteratif guna meminimalkan kesalahan dan meningkatkan efisiensi pengembangan sebagai berikut:

1) Proses dari menerima requirement - merancang *skema* DB – Coding - Testing - *Deployment - Server Development*

Alur dimulai dari analisis requirement HR (misalnya presensi geo-tagging), lalu rancang ERD dan *skema PostgreSQL*, coding controller/routes di *Express.js*, *testing* unit/integration via *Postman*, *deployment ke Docker container*, dan *monitoring server* untuk skalabilitas. Proses ini iteratif via *Scrum*, mengurangi bug dan memastikan reliabilitas produksi.

3.4 Deskripsi Project/Produk (dalam bentuk *diagram* dan uraian)

Naraworks HRIS merupakan sistem *backend* berbasis *Express.js* yang mengintegrasikan modul presensi dan HRM dengan penggajian untuk PT

Naramedic, dirancang dengan arsitektur *client-server-database* menggunakan *PostgreSQL*

3.4.1 Gambaran Umum Sistem Presensi & HRM

a. Penjelasan singkat fungsi sistem (mencatat kehadiran, izin, lembur dan integrasi gaji)

Sistem mencatat presensi harian, lembur, cuti, dan kunjungan kerja secara *real-time* dengan validasi geo-tagging, serta mengintegrasikan data *ke* modul *payroll* untuk perhitungan gaji otomatis termasuk tunjangan lembur.

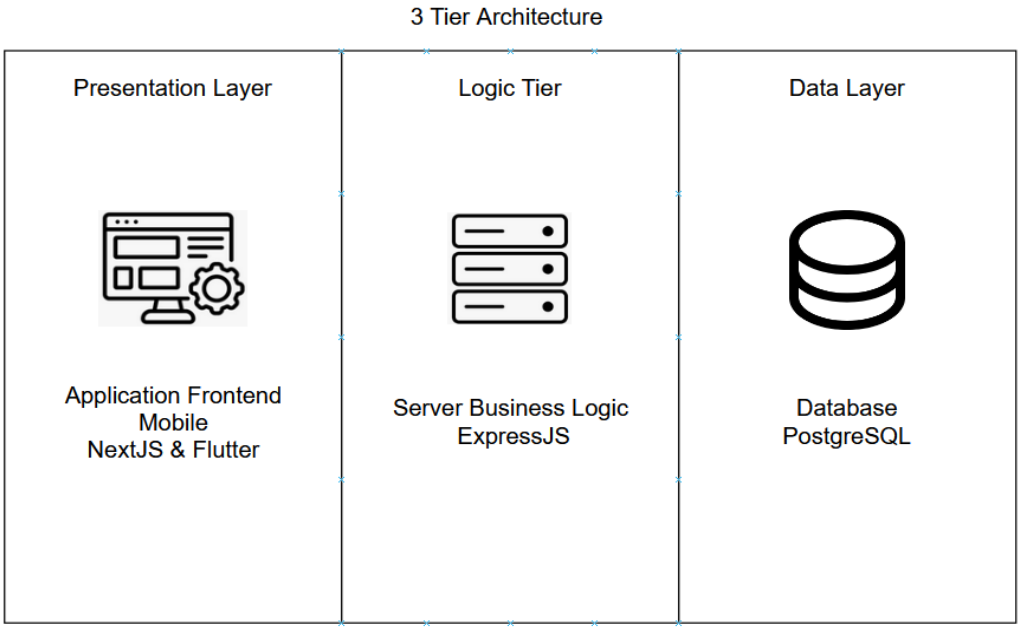
b. Target pengguna (karyawan dan admin HR)

Target utama adalah karyawan untuk *check*-in/out dan pengajuan izin/lembur, serta admin HR untuk approve *request*, *monitoring* attendance, dan generate laporan *payroll*.

3.4.2 Arsitektur Sistem (*Diagram*)

Arsitektur sistem adalah representasi *visual* atau deskriptif dari komponen-komponen yang membentuk sebuah sistem, serta bagaimana komponen tersebut saling berinteraksi satu sama lain. Arsitektur ini berfungsi sebagai peta jalan (blueprint) dalam pengembangan perangkat lunak.

a. *Diagram* Arsitektur (*client* - *server* - *database*)Sistem ini menggunakan 3-Tier Architecture, yang memisahkan fungsi sistem *ke* dalam tiga lapisan (layer) utama yang saling terhubung. Visualisasi dari arsitektur ini dapat dilihat pada Gambar 3.1.



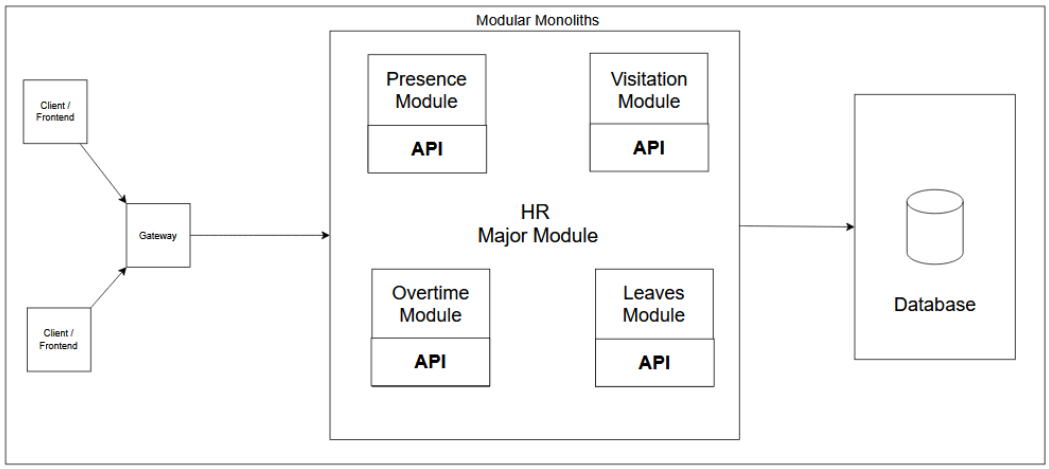
Gambar 3. 1 Gambar 3 Tier Architecture

1) Presentation Tier (*Client*) Ini adalah lapisan paling atas yang berinteraksi langsung dengan pengguna.
   a) Fungsi: Menampilkan antarmuka pengguna (UI) dan menerima *input* dari user.
   b) Media: Browser (*Chrome*, Edge) atau aplikasi *mobile*.
   c) Teknologi: HTML, CSS, *JavaScript*, atau React/Vue.
2) *Application* Tier (*Server*/Logic) Lapisan ini berada di tengah dan berfungsi sebagai "otak" dari sistem.
   a) Fungsi: Memproses logika bisnis, melakukan perhitungan, dan menjembatani komunikasi antara *Client* dan *Database*.
   b) Teknologi: PHP (Laravel), *Node.js*, Python (Django), atau Java.
3) Data Tier (*Database*) Lapisan ini adalah tempat penyimpanan data secara permanen.
   a) Fungsi: Menyimpan, mengambil, dan mengelola data yang dikirimkan melalui *Application* Tier.
   b) Teknologi: MySQL, *PostgreSQL*, atau SQL *Server*.

b. Modular Monolith Architecture Text

Modular Monolith adalah sebuah pendekatan arsitektur di mana seluruh kode program berada dalam satu unit penyebaran (single *deployment* unit), namun secara internal dibagi menjadi modul-modul yang terpisah dan

independen berdasarkan fungsi bisnisnya. Diagram arsitektur ini ditunjukkan pada Gambar 3.2.



Gambar 3. 2 Modular Monoliths Architecture Diagram (Sumber : Hasil buatan sendiri)

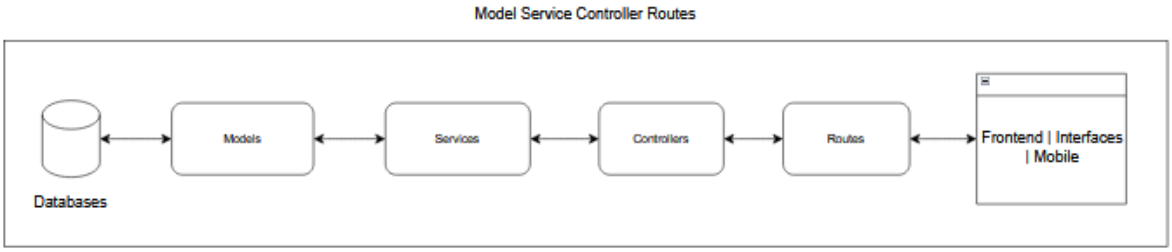
Berbeda dengan monolith tradisional yang kodenya sering kali tercampur (spaghetti *code*), Modular Monolith memaksa setiap modul untuk memiliki batasan yang jelas (bounded context).

"A modular monolith is a software design pattern where the application is *structured* as a single deployable unit but *compose*d of distinct, independent modules that communicate through well-defined interfaces" (Newman, 2021).

c. *Code* Architecture (*Model Service Controller Routes Code Architecture*)

Arsitektur *Model–Service–Controller–Routes* merupakan pengembangan dari pola *Model–View–Controller* (MVC) dengan menambahkan lapisan Service sebagai pemisah logika bisnis dari controller. Pada arsitektur ini, Model berfungsi untuk mengelola *struktur* dan interaksi data dengan *database*, sedangkan *Control*ler bertugas menerima dan memproses permintaan dari *client* melalui Routes. Lapisan Service berperan sebagai pusat logika bisnis, seperti validasi data, perhitungan, dan pengolahan proses inti sistem, sehingga controller hanya berfokus pada pengaturan alur *request* dan response. Dengan pemisahan tanggung jawab yang jelas ini,

struktur kode menjadi lebih terorganisir, mudah dipelihara, serta memudahkan pengembangan dan pengujian sistem di masa mendatang.



Gambar 3. 3 Model Service Controller Routes Architecture Diagram (Sumber : Hasil Buatan Sendiri)

d. *Code* Folder Architecture

Tabel 3.2 menjelaskan arsitektur folder kode yang digunakan dalam pengembangan sistem dengan pendekatan *Model–Service–Controller–Routes*. Setiap komponen memiliki peran yang berbeda namun saling berkaitan dalam menjalankan alur sistem. *Routes* berfungsi untuk menentukan alamat URL serta mengarahkan permintaan (*request*) *ke Controller* yang sesuai. *Controller* berperan sebagai penghubung antara permintaan dari pengguna dan proses bisnis di dalam sistem. *Service* bertanggung jawab dalam mengolah logika bisnis utama, sedangkan *Model* berfungsi sebagai pengelola data dan interaksi dengan *database*. Penggunaan analogi *rest*oran pada tabel ini bertujuan untuk mempermudah pemahaman konsep pembagian peran antar komponen dalam arsitektur kode.

Tabel 3. 2 Tabel Code Folder Architecture

Component	**Role**
Routes	Menentukan alamat URL dan mengarahkan permintaan (*request*) *ke Control*ler yang tepat.
*Control*ler	**Pelayan Restoran:** Mencatat pesanan pelanggan dan memberikan makanan yang sudah jadi

Service	**Koki Dapur:** Orang yang benar-benar memasak dan mengolah bahan sesuai resep.
Model	**Buku Stok/Gudang:** Catatan inventaris bahan makanan yang tersedia di gudang.

(Sumber : Hasil Olahan Sendiri)

3.4.3 *Use Case Diagram*

Untuk memahami interaksi antara pengguna dan sistem secara menyeluruh, diperlukan sebuah pemetaan fungsional yang menggambarkan batasan serta hak akses dari setiap entitas yang terlibat. Visualisasi ini dituangkan dalam bentuk *Use Case Diagram* yang merepresentasikan proses bisnis inti pada sistem HRIS Naraworks.

a. *Use Case Diagram*

Sistem Manajemen Presensi, Visitasi, dan Lembur Karyawan

Gambar 3. 4 HR and Payroll Sistem Use Case Diagram (Sumber: Hasil Olahan Sendiri)

Gambar 3.4 ini menggambarkan interaksi antara aktor dan sistem dalam pengelolaan *Human Resource Management* (HRM) dan *payroll*. Terdapat tiga aktor utama pada sistem, yaitu **S*up*eradmin**, **Admin**, dan ***User*s**, yang masing-masing memiliki hak akses dan fungsi berbeda sesuai dengan perannya. *Diagram* ini bertujuan untuk menunjukkan batasan sistem serta fitur-fitur yang dapat diakses oleh setiap aktor.

b. Deskripsi singkat aktor yang terlibat

Dalam pengoperasian sistem HRIS Naraworks, terdapat pembagian peran yang jelas guna menjaga keamanan data dan efisiensi alur kerja. Identifikasi aktor tersebut bertujuan untuk mendefinisikan tanggung jawab dan batasan akses sebagai berikut:

1) S*up*eradmin

S*up*eradmin memiliki hak akses penuh terhadap sistem, bisa mengakses semua *endpoint* yang ada terutama terhadap *management* data, dapat menambah dan menghapus admin, mengatur konfigurasi sistem. Namun s*up*eradmin tidak memiliki data employee karena tidak merupakan bagian user atau employee yang akan ikut *ke* dalam penghitungan *payroll* atau penggajian.

2) Admin

Admin memiliki hak akses terhadap berbagai *endpoint* dan fitur *management* data baik CREATE, READ, UPDATE dan *DELETE*. Tugas utama Admin sendiri adalah untuk dapat melakukan *management* data menghadapi segala kekurangan sistem apabila ada kekurangan dalam hal alur data. Melakukan persetujuan jika ada presensi karyawan yang tidak sesuai lokasi atau waktu hadir seharusnya. melakukan *management* data terhadap segala pengajuan, baik lemburan dan atau pengajuan cuti oleh karyawan.

3) *User*s

*User*s memiliki hak akses terbatas terhadap beberapa *endpoint* dan fitur yang ada. Endpoint dan fitur ini terdiri dari *endpoint* untuk melakukan kehadiran harian, kehadiran kunjungan kerja, kehadiran lemburan, pengajuan cuti, dan melihat segala data terkait users atau employee itu sendiri. Segala perubahan data di dalam aplikasi, tidak dapat dilakukan oleh users, hal itu harus dilakukan oleh s*up*eradmin atau admin.

3.4 Requirement/Spesifikasi Project/Produk

Keberhasilan pengembangan dan implementasi sistem HRIS Naraworks sangat bergantung pada ketersediaan sumber daya pendukung yang memadai. Bagian ini menjelaskan spesifikasi teknis yang menjadi standar minimum, baik dari sisi perangkat keras maupun perangkat lunak, guna menjamin stabilitas performa sistem di lingkungan pengembangan maupun produksi.

3.5.1 Kebutuhan Perangkat Keras (*Hardware*)

Perangkat keras berperan sebagai infrastruktur utama untuk menjalankan *server backend* dan basis data. Spesifikasi yang digunakan mencakup perangkat yang digunakan oleh pengembang untuk proses *coding* serta spesifikasi *server* untuk kebutuhan penyebaran (*deployment*):

a. Spesifikasi laptop pengembang
   i. Model : MSI P*rest*ige 14H B12UCX
   ii. *Process*or : Intel *Core* i5-12500H (12th Gen, 3.1 GHz)
   iii. RAM : 16 GB
   iv. GPU : Intel Iris Xe Graphics & NVIDIA GeForce RTX 2050
   v. Sistem Operasi : Windows 11 Home 64-bit

b. Spesifikasi *Server*

   Untuk *server*, pada pengembangan kali ini dipilih menggunakan VPS *Hosting* sebagai media penyimpanan seluruh *resource* sistem, berikut detail dari spesifikasi *server*.

Paket	: KVM 1 *Hosting*er
*Process*or	: 1 vCPU *core*
RAM	: 4 GB
Storage	: 50 GB NVMe disk space
Bandwith	: 4 TB
Biaya Layanan	: Rp. 1.162.800 / 12 bulan pertama
	Rp. 1.872.000 / 12 bulan berikutnya

3.5.2 Kebutuhan Perangkat Lunak (Software *Tools*)

Selain perangkat fisik, diperlukan serangkaian perangkat lunak dan kerangka kerja (*framework*) untuk membangun logika sistem. Pemilihan *tools* ini didasarkan pada kebutuhan skalabilitas, keamanan, dan kemudahan kolaborasi tim, yang meliputi:

a. Bahasa Pemrograman & *Framework* menggunakan NodeJS, *Express*JS.
b. *Database Management System* menggunakan sistem *relational database management* yaitu *PostgreSQL*
c. *API* Testing Tool menggunakan *postman*.
d. *Code* Editor menggunakan *Visual Studio Code*
e. *Version Control* menggunakan *Github* Desktop dan *Github Web*.

3.5.3 Kebutuhan Fungsional (*Backend* Focus)

Kebutuhan fungsional mendefinisikan layanan atau fitur spesifik yang harus disediakan oleh sistem agar tujuan bisnis dapat tercapai. Fokus utama pada bagian ini adalah memastikan seluruh proses manajemen sumber daya manusia, mulai dari autentikasi hingga pelaporan, dapat diolah oleh *backend* secara akurat: (Lanjutkan ke rincian poin a, b, c yang Anda miliki).

a. Authentication & Authorization

1) *User* Authentication
   a) Melakukan proses **login** menggunakan email dan *password*.
   b) Melakukan **validasi kredensial** terhadap data pengguna yang tersimpan di *database*.
   c) Menghasilkan **access token (*JWT*)** setelah autentikasi berhasil.

d) Menolak akses apabila email atau *password* tidak valid.
e) Mengelola ***session* berbasis token** dengan masa berlaku tertentu.
f) Menyediakan fitur **logout** dengan mekanisme invalidasi token

2) Role & Access *Control*
a) Mengelola **hak akses berdasarkan role pengguna** (S*up*eradmin, Admin, dan *User*s).
b) Membatasi akses *endpoint backend* sesuai role pengguna.
c) Mencegah pengguna mengakses fitur di luar kewenangannya.
d) Mengizinkan admin melakukan pengelolaan role pengguna.

b. Employee *Management*

1) Data Karyawan
a) Menyimpan data karyawan (nama, email, jabatan, departemen, status kerja).
b) Melakukan **CRUD (Create, Read, Update, Delete)** data karyawan.
c) Mengaitkan data karyawan dengan akun pengguna.
d) Menyediakan fitur pencarian dan filter data karyawan.
e) Menyimpan status aktif atau non-aktif karyawan.

c. *Attendance* & Presence *Management*

1) Presence (*Check*-In / *Check*-Out)
a) Mencatat waktu ***check*-in** dan ***check*-out** karyawan.
b) Memastikan karyawan hanya dapat *check*-in satu kali per hari.
c) Memvalidasi *check*-in berdasarkan status agreement lembur (jika ada).
d) Menyimpan data kehadiran beserta timestamp *server*.
e) Menolak *check*-in jika karyawan belum mendapatkan persetujuan lembur.

2) *Attendance* Summary
a) Menghasilkan ringkasan kehadiran harian, mingguan, dan bulanan.
b) Menghitung total jam kerja dan jam lembur karyawan.
c) Menyediakan data rekap kehadiran per karyawan dan per departemen.

d. Agreement & Approval *System*

1) Presence Agreement
    a) Mencatat pengajuan kehadiran khusus (lembur).
    b) Mengaitkan agreement kehadiran dengan data karyawan.
    c) Menyimpan status agreement (Pending, Approved, Rejected).
2) Approval *Work*flow
    a) Menyediakan alur persetujuan bertingkat (Employee → Manager → HR).
    b) Mencatat waktu dan pengguna yang memberikan persetujuan.
    c) Mengirimkan notifikasi status approval kepada karyawan.
    d) Menolak proses kehadiran jika agreement belum disetujui.

e. Leave *Management*

1) Leave Request
    a) Menerima pengajuan cuti dari karyawan.
    b) Menyimpan jenis cuti, durasi, dan alasan cuti.
    c) Memvalidasi ketersediaan kuota cuti.
2) Leave Approval
    a) Mengelola persetujuan cuti oleh atasan atau HR.
    b) Mengubah status cuti (Pending, Approved, Rejected).
    c) Mengirimkan notifikasi hasil persetujuan cuti.
3) Leave Quota *Management*
    a) Mengelola kuota cuti tahunan karyawan.
    b) Mengurangi kuota cuti secara otomatis setelah persetujuan.
    c) Menampilkan sisa kuota cuti karyawan.

f. Reporting & Export

1) *System* Settings
    a) Menghasilkan laporan kehadiran, cuti, dan *payroll*.
    b) Menyediakan laporan berdasarkan periode waktu.
    c) Mengelompokkan laporan berdasarkan karyawan atau departemen.
2) Tree/Hierarchy Settings
    a) Mengekspor laporan *ke form*at PDF dan Excel.

b) Mengatur hak akses pengguna terhadap fitur export.
c) Menyimpan histori export data.

b. *Error Handling* & Validation

1) Reports
   a) Melakukan validasi *input* pada setiap *endpoint API*.
   b) Mengembalikan response error yang konsisten (*HTTP* Status *Code*).
   c) Menyediakan pesan error yang in*form*atif.
   d) Mencatat error sistem *ke* dalam log *server*.
   e) Menangani error autentikasi dan otorisasi secara terpusat.

3.5.4 Kebutuhan Non Fungsional

a. Keamanan data (Enkripsi *password*, *JWT* Token).
   Sistem **harus memenuhi** kebutuhan keamanan data sebagai berikut:

1) Sistem **harus mengenkripsi *password* pengguna** menggunakan algoritma hashing yang aman (misalnya **bcrypt**) sebelum disimpan *ke* dalam *database*.
2) Sistem **tidak diperbolehkan menyimpan *password* dalam bentuk plaintext**.
3) Sistem **harus menggunakan JSON *Web* Token (*JWT*)** sebagai mekanisme autentikasi dan otorisasi pengguna.
4) *JWT* **harus memiliki masa berlaku (expiration time)** untuk mencegah penyalahgunaan token.
5) Sistem **harus melakukan validasi token *JWT*** pada setiap *endpoint* yang dilindungi.
6) Sistem **harus membatasi akses *API* berdasarkan role pengguna** (Role-Based Access *Control*).
7) Sistem **harus menggunakan *HTTP*S** pada lingkungan produksi untuk melindungi transmisi data.
8) Sistem **harus mencatat aktivitas autentikasi** (login, logout, gagal login) *ke* dalam sistem log.

Indikator keberhasilan:

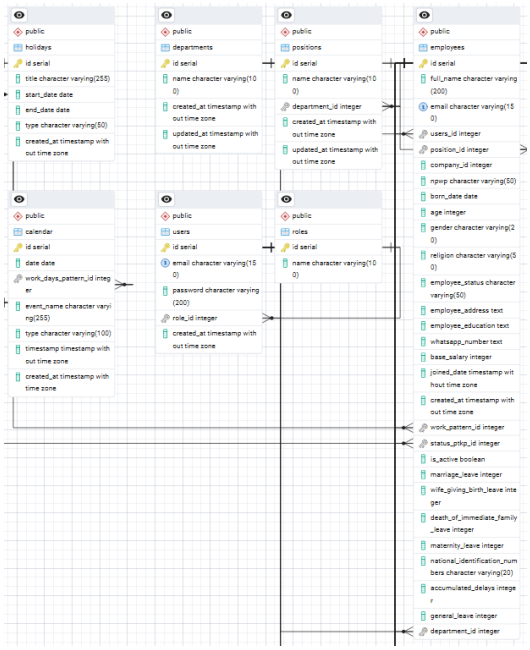
1) *Password* tersimpan dalam bentuk hash.
2) Endpoint terlindungi tidak dapat diakses tanpa token valid.
3) Token kadaluarsa tidak dapat digunakan kembali.
4) Kecepatan respon *API* (Per*form*ance).

3.6 Desain Project/Produk

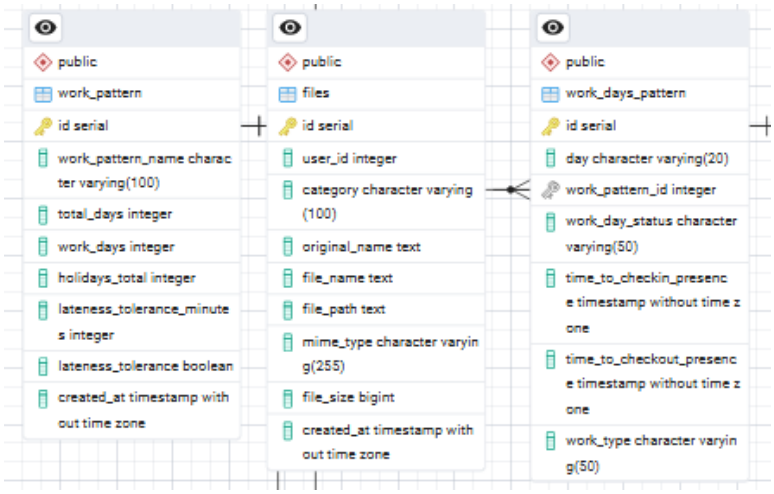
Perancangan basis data dalam penelitian ini digambarkan melalui *Entity Relationship Diagram* (ERD). Diagram ini memberikan gambaran menyeluruh mengenai entitas-entitas yang terlibat beserta atribut kunci yang menghubungkan satu tabel dengan tabel lainnya. *Struktur relasional* antar entitas tersebut dapat dilihat pada gambar berikut:

3.6.1 Perancangan Basis Data (ERD)

Gambar 3.5 dari rancangan basis data ini berfokus pada entitas utama penyusun organisasi, di mana tabel *employees* berfungsi sebagai pusat data yang menyimpan informasi biodata lengkap karyawan. Tabel ini terintegrasi secara langsung dengan tabel *users* dan *roles* untuk mengelola hak akses keamanan sistem, serta tabel *departments* dan *positions* yang mendefinisikan *struktur* jabatan karyawan dalam perusahaan. Selain itu, terdapat tabel *holidays* dan *calendar* yang berfungsi sebagai acuan waktu operasional dan hari libur guna memastikan sinkronisasi jadwal kerja di seluruh sistem.



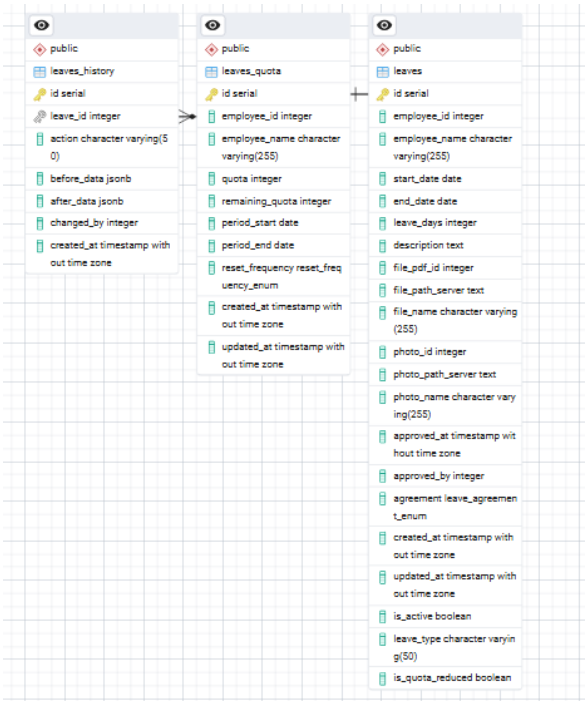
Gambar 3. 5 ERD Table Employees, Positions (Sumber: Hasil Olahan Sendiri)



Gambar 3. 6 ERD Work Days Pattern, Files, Work Pattern

(Sumber: Hasil Olahan Sendiri)

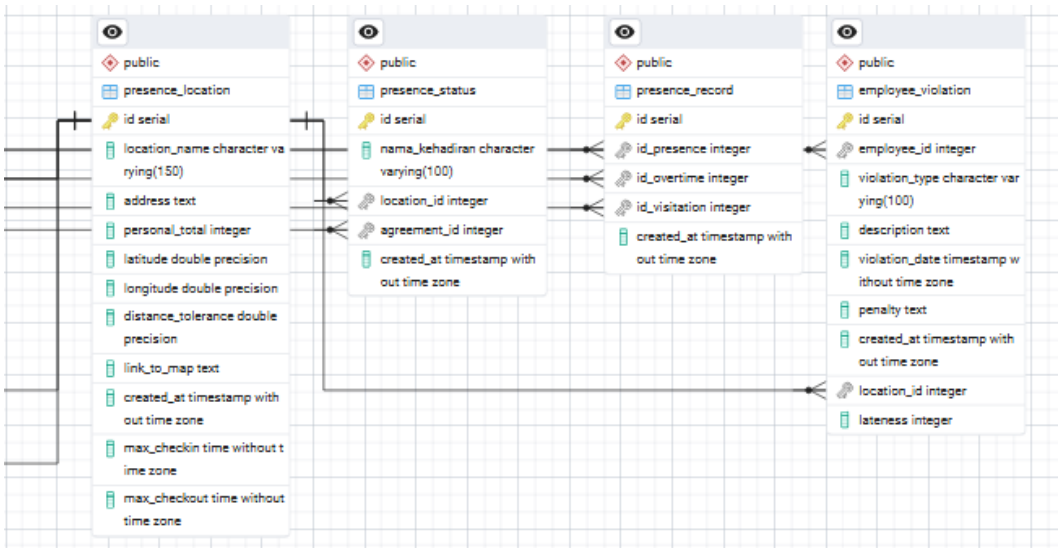
Gambar 3.6 ini mengatur aspek jam kerja dan penyimpanan berkas melalui tabel work_pattern serta work_days_pattern, yang merinci jadwal shift, waktu *check-in/out*, dan batas toleransi keterlambatan. Di sisi lain, untuk mendukung kebutuhan dokumentasi digital, tabel files bertindak sebagai repositori metadata yang menyimpan informasi lengkap mengenai setiap dokumen yang diunggah *ke* dalam sistem, seperti nama file, lokasi penyimpanan (*path*), hingga tipe dokumen. Hal ini memastikan bahwa seluruh lampiran pendukung, baik untuk data karyawan maupun pengajuan cuti, terorganisir dengan baik dalam basis data.



Gambar 3. 7 ERD Table Leaves, Leaves Quota, Leaves History

(Sumber: Hasil Olahan Sendiri)

Gambar 3.7 ini rancangan basis data berfokus pada pengelolaan absensi dan hak cuti karyawan melalui tabel leaves yang mencatat detail pengajuan mulai dari tanggal hingga alasan. Untuk menjaga akurasi administrasi, tabel leaves_quota digunakan untuk memantau sisa kuota cuti karyawan secara *real-time* dalam periode tertentu. Seluruh aktivitas atau perubahan pada data cuti direkam dalam tabel leaves_history, sehingga riwayat persetujuan atau penolakan pengajuan dapat terlacak dengan transparan bagi pihak manajemen maupun karyawan.



Gambar 3. 8 ERD Employee Visitatoin, Presence Record – (Sumber: Hasil Olahan Sendiri)

Gambar 3.8 menjelaskan mekanisme validasi kehadiran yang sangat bergantung pada tabel presence_location untuk menyimpan koordinat geografis dan radius toleransi lokasi kerja. Data transaksi kehadiran harian dicatat pada tabel presence_record, yang bekerja sama dengan presence_status untuk menentukan klasifikasi status kehadiran pengguna. Apabila terdapat ketidaksesuaian dengan aturan perusahaan, tabel employee_violation akan merekam data pelanggaran tersebut, termasuk informasi keterlambatan dan penalti yang terkait dengan lokasi presensi tertentu.

3.6.2 *Struktur* Tabel (*Schema Database*)

Struktur tabel atau database *schema* dalam sistem ini dirancang sedemikian rupa untuk menjamin integritas dan efisiensi akses data. Penjelasan mengenai komponen data tersebut akan dipaparkan melalui tabel-tabel berikut, yang mencakup informasi dasar pada Employees Table, pengaturan hak akses pada Users Table, serta data

spasial yang tersimpan pada Presence Location Table. Penjelasan detail mengenai atribut, tipe data, dan keterangan dari masing-masing tabel tersebut dapat dilihat pada bagian di bawah ini.

Tabel 3. 3 Employees Table

Nama Kolom	Tipe Data	Keterangan
id	integer	ID unik karyawan
full_name	varchar(200)	Nama lengkap
email	varchar(150)	Email karyawan
users_id	integer	Relasi *ke* tabel users
position_id	integer	Relasi *ke* tabel posisi
company_id	integer	Relasi *ke* tabel perusahaan
npwp	varchar(50)	Nomor NPWP
born_date	date	Tanggal lahir
age	integer	Usia
gender	varchar(20)	Jenis kelamin
religion	varchar(50)	Agama
employee_status	varchar(50)	Status karyawan
employee_address	text	Alamat
employee_education	text	Pendidikan
whatsapp_number	text	Nomor WhatsApp
base_salary	integer	Gaji pokok
joined_date	timestamp	Tanggal bergabung
created_at	timestamp	Waktu dibuat
*work*_pattern_id	integer	Relasi *ke* pola kerja
status_ptkp_id	integer	Relasi *ke* status PTKP
is_active	boolean	Status aktif

marriage_leave	integer	Cuti menikah
wife_giving_birth_leave	integer	Cuti istri melahirkan
death_of_immediate_family_leave	integer	Cuti keluarga meninggal
maternity_leave	integer	Cuti melahirkan
national_identification_numbers	varchar(20)	Nomor NIK
accumulated_delays	integer	Akumulasi keterlambatan
general_leave	integer	Cuti umum
department_id	integer	Relasi *ke* departemen

(Sumber: Hasil Olahan Sendiri)

Tabel 3.3 berfungsi sebagai entitas utama yang menyimpan informasi komprehensif mengenai profil setiap tenaga kerja di PT Naramedic. Tabel ini mencakup berbagai atribut penting mulai dari identitas pribadi (nama, email, NIK), informasi demografis (tanggal lahir, jenis kelamin, agama), hingga data administratif kepegawaian seperti gaji pokok, status PTKP, departemen, dan akumulasi keterlambatan. Selain itu, tabel ini juga mengelola jatah cuti karyawan secara spesifik untuk berbagai keperluan, sehingga menjadi sumber data *primer* bagi sistem dalam melakukan perhitungan penggajian dan manajemen sumber daya manusia secara otomatis.

Tabel 3. 4 Users Tabel

Nama Kolom	Tipe Data	Keterangan
id	integer	ID unik user
email	varchar(150)	Email user (wajib diisi)
password	varchar(200)	*Password* user
role_id	integer	Relasi *ke* tabel role
created_at	timestamp	Waktu pembuatan data

(Sumber: Hasil Olahan Sendiri)

Tabel 3.4 dirancang khusus untuk menangani aspek keamanan dan autentikasi akses *ke* dalam sistem Naraworks. Tabel ini menyimpan kredensial login berupa alamat email dan kata sandi yang telah dienkripsi, serta menyimpan informasi peran

(*role_id*) untuk menentukan batasan hak akses pengguna di dalam aplikasi. Dengan adanya relasi antara tabel ini melalui *id* pengguna, sistem dapat memastikan bahwa setiap aktivitas yang dilakukan di dalam platform dapat dipertanggungjawabkan dan sesuai dengan kewenangan yang diberikan kepada masing-masing individu.

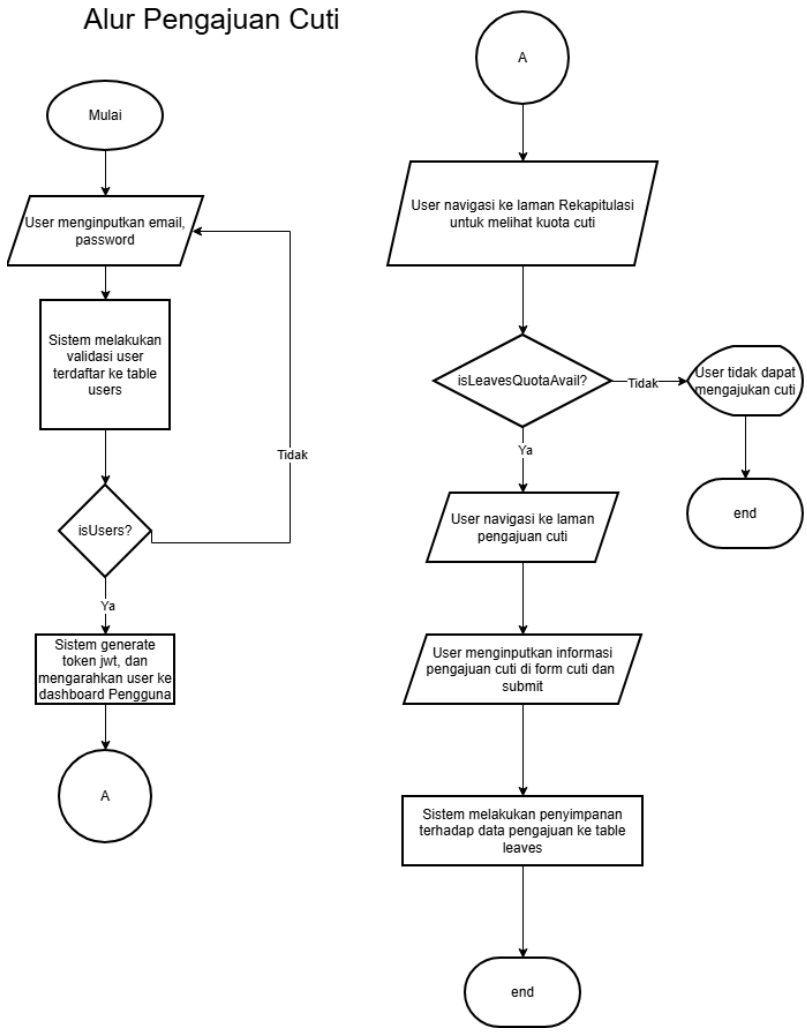
Tabel 3. 5 Table Presence Location

Nama Kolom	Tipe Data	Keterangan
id	integer	ID lokasi absensi
location_name	varchar(150)	Nama lokasi
address	text	Alamat lokasi
personal_total	integer	Jumlah personil di lokasi
latitude	double precision	Koordinat latitude
long*it*ude	double precision	Koordinat long*it*ude
distance_tolerance	double precision	Toleransi jarak absensi (meter)
link_to_map	text	Link *ke* peta lokasi
created_at	timestamp	Waktu pembuatan data
max_*check*in	time	Batas maksimal *check*-in
max_*check*out	time	Batas maksimal *check*-out

(Sumber: Hasil Olahan Sendiri)

Tabel *3.5* merupakan tabel referensi yang menyimpan parameter geografis untuk memvalidasi presensi karyawan berbasis lokasi (Geofencing). Tabel ini menyimpan data titik koordinat presisi berupa *latitude* dan *longitude* serta *distance_tolerance* untuk menentukan radius area yang diizinkan bagi karyawan untuk melakukan absensi. Selain koordinat fisik, tabel ini juga mengatur kebijakan waktu operasional melalui kolom *max_checkin* dan *max_checkout*, yang berfungsi sebagai standar pembatas jam kerja guna menjaga kedisiplinan dan akurasi pencatatan kehadiran karyawan di setiap lokasi kerja.

3.6.3 Perancangan Alur Logika (*Flowchart*)



Gambar 3. 9 Flowchart Alur Pengajuan Cuti (Sumber: Hasil Olahan Sendiri)

Penjelasan langkah-langkah logika dari *Request* masuk hingga *Response* keluar.

a. *User* membuka aplikasi/sistem, dan memasukkan kredensial ber*up*a Email dan *Password*.
b. Sistem melakukan pengecekan data yang di*input*kan terhadap data yang ada dalam *database* (table users), apakah data tersebut merupakan salah satu user yang terdaftar pada sistem.
c. Jika bukan merupakan user, maka akan kembali *ke* langkah (a).
d. Jika merupakan *user* terdaftar, maka sistem akan generate token untuk user dapat mengakses dashboard pengguna.

e. *User* melakukan navigasi *ke* laman Rekapitulasi untuk melihat kuota cuti *user.*
f. Pada laman rekapitulasi tersebut, user akan melihat kuota cuti yang ia miliki. Apabila ia tidak memiliki kuota tersisa, maka ia tidak dapat melakukan pengajuan.
g. Jika ia masih memiliki kuota tersisa maka *user* akan navigasi *ke* laman pengajuan cuti.
h. Selanjutnya *user* akan meng*input*kan informasi pengajuan cuti di *form* cuti dan melakukan submit
i. Sistem akan melakukan penyimpanan terhadap data pengajuan *ke* table cuti (leaves) untuk kemudian di alur yang berbeda akan di tindak lanjuti oleh admin.

3.6.4 Perancangan Api (*API* Contract / Endpoint)

Sebagai penghubung antara antarmuka pengguna (*front-end*) dan pusat data, perancangan *Application Programming Interface* (API) menjadi krusial untuk memastikan pertukaran data berjalan secara konsisten dan aman. Bagian ini mendefinisikan kontrak komunikasi teknis yang mencakup metode HTTP, jalur *endpoint*, serta fungsi spesifik dari setiap layanan yang disediakan oleh sistem. Spesifikasi lengkap mengenai jalur akses dan logika bisnis dari setiap layanan *backend* dirangkum dalam **Tabel 3.6** di bawah ini.

Tabel 3. 6 API Contract Table

Nama *API*	Method	Endpoint	Fungsi
Auth Register with Hashed *Password*	*POST*	/auth/register	Mendaftarkan akun pengguna baru dengan penyimpanan *password* yang telah di-hash secara aman pada

			sistem autentikasi. Dikhususkan untuk akun user pertama dan akan bertindak sebagai role *Superadmin*
Auth Login	*POST*	/auth/login	Melakukan proses autentikasi pengguna dan menghasilkan access token (*JWT*) untuk mengakses *resource* yang dilindungi
Auth Regist *User*/Employee	*POST*	users/v2	Membuat akun pengguna sekaligus data karyawan dan mengaitkannya dengan role serta informasi inisiasi pribadi karyawan dan organisasi.
Auth Set *Password First* Time	*POST*	/set-*password*	Mengatur *password* pertama kali bagi pengguna yang

			dibuat oleh admin atau HR sebelum login pertama. Akses terhadap *endpoint* ini diharuskan diakses menggunakan link yang berisi token dari email pengguna.
Get All *User*s Presence	*GET*	/employee-presences/:employeeId	Mengambil seluruh riwayat kehadiran (*check*-in dan *check*-out) karyawan berdasarkan ID karyawan.
Check-In Employee Presence	*POST*	/employee-presences-v2/*check*in	Mencatat waktu *check*-in karyawan pada sistem kehadiran dengan validasi aturan kehadiran.
Check-Out Employee Presence	*POST*	/employee-presences-v2/*check*out	Mencatat waktu *check*-out karyawan dan menghitung

			durasi jam kerja harian.
Get All *User* Visitation	*GET*	/employee-visitations/:employeeId	Mengambil data kunjungan kerja (visitation) karyawan berdasarkan ID karyawan.
Check-In Employee Visitation	*POST*	/employee-visitations/*check*in	Mencatat waktu *check*-in karyawan untuk aktivitas kunjungan kerja di luar kantor.
Check-Out Employee Visitation	*POST*	/employee-visitations/*check*out	Mencatat waktu *check*-out karyawan dari aktivitas kunjungan kerja.
Get All *User Overtime*	*GET*	/employee-overtimes/:employeeId	Mengambil seluruh data lembur karyawan berdasarkan ID karyawan.
Overtime Issueance	*POST*	/employee-overtimes	Mengajukan permohonan lembur oleh karyawan atau HR dengan detail

			waktu dan alasan lembur.
Overtime Issueance Common Update (Agree/Reject)	*PUT*	/employee-overtimes/:id	Memproses persetujuan atau penolakan pengajuan lembur oleh pihak berwenang.
Check-In Employee *Overtime*	*POST*	/employee-overtimes/*check*-in	Mencatat waktu mulai lembur karyawan setelah pengajuan lembur disetujui.
Check-Out Employee *Overtime*	*POST*	/employee-overtimes/*check*-out	Mencatat waktu selesai lembur dan menghitung total durasi lembur karyawan.
Leaves Issuance	*POST*	/leaves	Mengajukan permohonan cuti karyawan beserta jenis, durasi, dan alasan cuti
Leaves Issuance Common Update (Agree/Reject)	*PUT*	/leaves/:id	Memproses persetujuan atau penolakan pengajuan cuti

			oleh atasan atau HR.
Add Holidays to Automated Employee Presence	*POST*	/holidays	Menambahkan data hari libur nasional atau perusahaan yang akan digunakan dalam perhitungan kehadiran otomatis.
Get Dailly *Attendance* (Filtered by Date and Type) Join 3 Table (Presence	*GET*	/employee-attendance?date=2025-12-12	Mengambil rekap kehadiran harian karyawan berdasarkan tanggal dengan menggabungkan data presence, lembur, dan visitation.
Update MultiColumn on Multi table	*PUT*	/presence/multi-*up*date	Melakukan pembaruan beberapa kolom pada lebih dari satu tabel kehadiran secara atomik untuk keperluan koreksi data.

(Sumber: Hasil Olahan Sendiri)

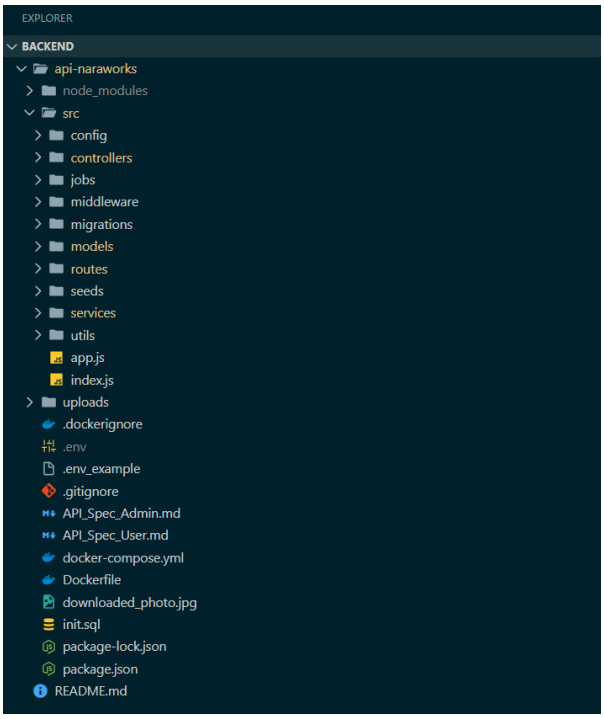
Tabel 3.6 ini mengimplementasikan standar RESTful API yang terbagi ke dalam beberapa modul utama, yaitu autentikasi, manajemen kehadiran, lembur, dan pengajuan cuti. Pada modul autentikasi, penggunaan metode **POST** pada *endpoint* /auth/register dan /auth/login menjamin keamanan data sensitif dengan penerapan *hashing password* dan penggunaan *JSON Web Token* (JWT) untuk akses sumber daya. Selain itu, **Tabel 3.6** juga menunjukkan fleksibilitas sistem dalam menangani data kehadiran yang kompleks, di mana terdapat pemisahan antara kehadiran reguler, kunjungan kerja (*visitation*), dan lembur guna menghasilkan perhitungan jam kerja yang akurat. Integrasi data ini dipertegas pada *endpoint* /employee-attendance yang mampu melakukan proses *join* dari tiga tabel utama untuk menyajikan rekap harian yang komprehensif bagi pihak manajemen.

3.7 Implementasi

Tahap implementasi merupakan fase di mana seluruh rancangan teknis, mulai dari basis data hingga kontrak API, diwujudkan ke dalam bentuk kode program yang fungsional. Pada tahap ini, fokus utama terletak pada penerjemahan logika bisnis ke dalam struktur perangkat lunak yang andal, efisien, dan mudah untuk dikelola. Implementasi ini mencakup penyusunan arsitektur kode, integrasi sistem autentikasi, serta pembangunan layanan inti untuk manajemen sumber daya manusia.

3.7.1 *Struktur* Direktori Proyek

Struktur direktori dalam proyek ini dirancang menggunakan pola organisasi yang sistematis untuk mendukung prinsip Separation of Concerns (SoC). Hal ini dilakukan agar setiap komponen kode memiliki tanggung jawab yang spesifik, sehingga memudahkan proses pemeliharaan (maintenance) dan pengembangan fitur di masa mendatang. Berikut adalah penjelasan mengenai fungsi dari masing-masing folder utama dalam proyek:



Gambar 3. 9 Gambar Susunan Folder Project

Berikut adalah penjelasan mengenai fungsi dari masing-masing folder utama dalam proyek berdasarkan susunan pada **Gambar 3.10**:

a. Folder Routes: Bertindak sebagai gerbang utama atau pendefinisian alamat URL (endpoint) aplikasi. Folder ini bertanggung jawab untuk memetakan setiap permintaan *HTTP* (*GET*, *POST*, *PUT*, *DELETE*) dari klien *ke* fungsi pengontrol (controller) yang sesuai.

b. Folder Controller: Berfungsi sebagai jembatan komunikasi antara logika bisnis dan permintaan pengguna. Folder ini menerima data dari routes, memvalidasi *input* dasar, dan memanggil layanan (service) yang diperlukan untuk kemudian mengembalikan respon kepada pengguna dalam *form*at yang sesuai (seperti JSON).

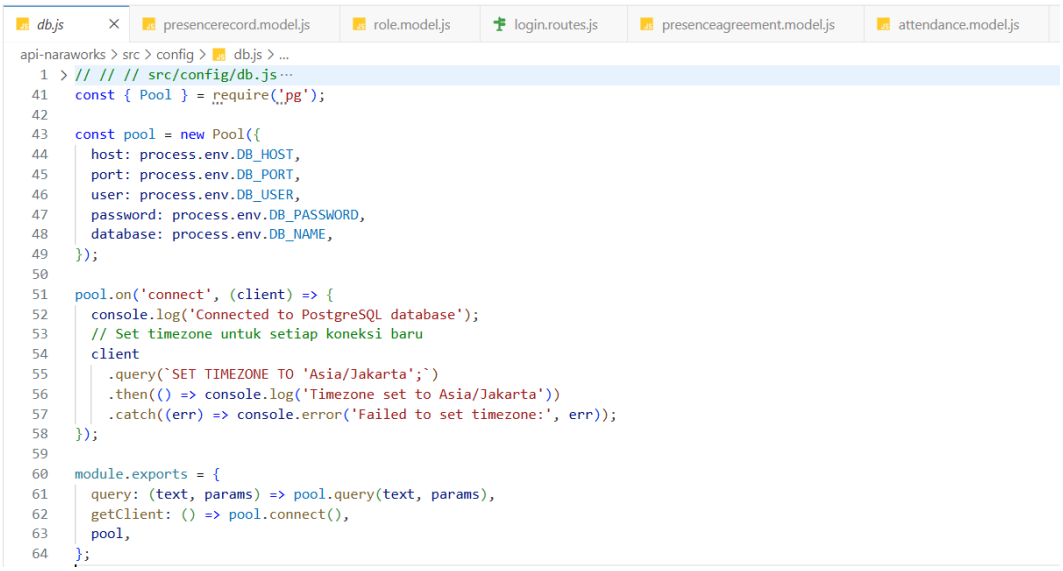
c. Folder Service: Merupakan lapisan tempat seluruh logika bisnis utama (business logic) berada. Folder ini menangani pemrosesan data yang kompleks, melakukan perhitungan, serta mengoordinasikan interaksi antar berbagai model sebelum akhirnya mengirimkan hasil pemrosesan kembali *ke* controller.

d. Folder Models: Bertanggung jawab atas pengelolaan data dan interaksi langsung dengan basis data. Folder ini mendefinisikan *skema* tabel, tipe data, serta relasi antar entitas (seperti yang telah dijelaskan pada bagian ERD) untuk memastikan integritas data tetap terjaga.

e. Folder Utils (Utilities): Berisi kumpulan fungsi pembantu atau *pustaka* kecil yang bersifat umum dan sering digunakan berulang kali di berbagai bagian aplikasi. Contoh isinya meliputi fungsi pem*forma*tan tanggal, enkripsi *password*, konfigurasi pengiriman email, atau helper untuk validasi tertentu.

3.7.2 Implementasi Konfigurasi *Database*

Tahap awal dalam implementasi perangkat lunak adalah membangun koneksi yang stabil antara aplikasi dan sistem manajemen basis data. Konfigurasi ini sangat krusial karena seluruh transaksi data, mulai dari penyimpanan informasi karyawan hingga pencatatan kehadiran, bergantung pada efisiensi *pool* koneksi yang dibentuk. Implementasi teknis mengenai pengaturan parameter *database* dan penyesuaian zona waktu pada sistem ini dapat dilihat pada **Gambar 3.11**.



Gambar 3.11 Gambar Potongan Kode Koneksi Database (Sumber: Hasil Olahan Sendiri)

Berdasarkan **Gambar 3.11**, aplikasi menggunakan pustaka pg untuk mengelola koneksi ke PostgreSQL melalui objek Pool. Keamanan data dijaga dengan menggunakan variabel lingkungan (*environment variables*) seperti DB_HOST, DB_USER, dan DB_*PASSWORD* untuk menghindari penulisan kredensial secara langsung di dalam kode. Selain itu, terdapat logika otomatis untuk mengatur zona

waktu menjadi 'Asia/Jakarta' setiap kali koneksi baru terbentuk, guna menjamin sinkronisasi waktu yang akurat antara server dan basis data.

3.7.3 Implementasi *Backend* Logic (*Control*ler/Service)

Pada bagian ini, fokus utama adalah implementasi logika bisnis yang menangani validasi data dan alur kerja aplikasi. Salah satu fitur kritikal yang dikembangkan adalah sistem *geofencing* untuk memastikan validitas lokasi presensi karyawan.

```
// Ambil lokasi valid berdasarkan koordinat
async function getIsValidLocation(latitude, longitude) {
  if (typeof latitude === 'undefined' || typeof longitude === 'undefined') return null;

  await ensureOutsideLocationExists();

  const { rows: locations } = await query(`SELECT * FROM presence_location`);
  let validLocation = null;

  for (const loc of locations) {
    const distance = calculateDistance(latitude, longitude, loc.latitude, loc.longitude);
    if (distance <= loc.distance_tolerance) {
      validLocation = loc;
      break;
    }
  }

  // fallback ke lokasi default jika tidak ada yang cocok
  if (!validLocation) validLocation = { id: OUTSIDE_LOCATION_ID, name: 'Outside Location' };
  return validLocation;
}
```

Gambar 3. 10 Gambar Potongan Kode GetValidLocation (Sumber: Hasil Olahan Sendiri)

ditunjukkan pada **Gambar 3.12**. Fungsi ini bekerja dengan cara mengambil seluruh data lokasi yang terdaftar di basis data, kemudian melakukan iterasi untuk menghitung jarak antara koordinat pengguna dengan koordinat kantor menggunakan algoritma **Haversine**. Jika jarak berada di bawah ambang batas distance_tolerance, maka lokasi dianggap valid.

```
/* ----------------------------------------------------------------
   📍 Location Validation Logic
---------------------------------------------------------------- */
const OUTSIDE_LOCATION_ID = 999;

// Pastikan lokasi default "Outside" tersedia
async function ensureOutsideLocationExists() {
  const { rows } = await query(
    'SELECT id FROM presence_location WHERE id = $1',
    [OUTSIDE_LOCATION_ID]
  );

  if (rows.length === 0) {
    await query(
      `INSERT INTO presence_location
        (id, location_name, latitude, longitude, distance_tolerance, link_to_map, created_at, max_checkin, max_checkout)
       VALUES ($1, $2, $3, $4, $5, $6, NOW(), $7, $8)
       ON CONFLICT (id) DO NOTHING`,
      [OUTSIDE_LOCATION_ID, 'Outside Location', 0, 0, 0, null, '09:00:00', '21:00:00']
    );
  }
```

Gambar 3. 13 Gambar Potongan Kode Ensuring Outside Location (Sumber: Hasil Olahan Sendiri)

Untuk menjaga integritas sistem jika pengguna berada di luar jangkauan, terdapat fungsi ensureOutsideLocationExists seperti yang terlihat pada **Gambar 3.13**. Fungsi ini memastikan entitas lokasi "Outside" dengan ID khusus (999) selalu tersedia di tabel presence_location sebagai *fallback* otomatis.

api-naraworks > src > utils > haversine.js > ...

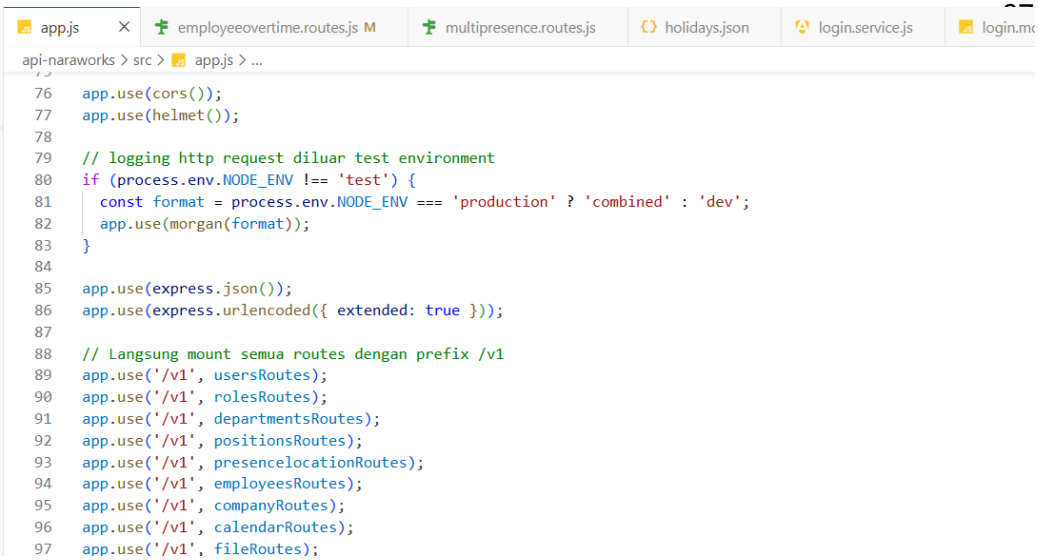
```
// haversine.js
function toRad(deg) {
  return deg * Math.PI / 180;
}

const radius = 20;

function haversine(lat1, lon1, lat2, lon2) {


  const R = 6371000; // meter
  const φ1 = toRad(lat1);
  const φ2 = toRad(lat2);
  const Δφ = toRad(lat2 - lat1);
  const Δλ = toRad(lon2 - lon1);

  const a = Math.sin(Δφ / 2) ** 2 +
            Math.cos(φ1) * Math.cos(φ2) *
            Math.sin(Δλ / 2) ** 2;

  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
  return R * c;
}

const dMeters = haversine(-6.35820356157271, 106.8228314250927, -6.358165575195065, 106.8229407251092); // Jakarta → Bandung (perkiraan)
if(dMeters <= radius) {
  console.log("lokasi valid");
}else{
  console.log("lokasi tidak valid");
}
console.log(`Jarak user terhadap kantor adalah ${dMeters.toFixed(0)} meter`); // output mis. "xxxxx meter"

module.exports = haversine;
```

Gambar 3. 14 Gambar Potongan Kode Haversine Code (Sumber: Hasil Olahan Sendiri)

algoritma **Haversine**. Fungsi haversine menerima parameter berupa garis lintang (*latitude*) dan garis bujur (*longitude*) dari dua titik, kemudian menghitung jarak geometrisnya berdasarkan jari-jari bumi (6.371.000 meter). Hasil perhitungan ini kemudian dikembalikan dalam satuan meter untuk divalidasi oleh sistem terhadap toleransi jarak yang diizinka

3.7.4 Implementasi *Routing API*

Implementasi rute (*routing*) merupakan tahap akhir dalam pembangunan *backend* yang berfungsi untuk memetakan permintaan HTTP dari klien ke fungsi *controller* yang tepat. Dengan struktur *routing* yang terorganisir, aplikasi dapat mengelola berbagai titik akses (*endpoints*) secara modular dan mudah dipelihara. Pemetaan rute utama dalam sistem ini ditunjukkan pada **Gambar 3.15**.



Gambar 3. 15 Gambar Potongan Kode Endpoint (Sumber: Hasil Olahan Sendiri)

Berdasarkan **Gambar 3.15**, sistem menggunakan *router* utama yang dipasang pada aplikasi dengan prefiks /v1. Hal ini dilakukan untuk mengelompokkan berbagai layanan berdasarkan modulnya, seperti usersRoutes, employeesRoutes, hingga calendarRoutes, sehingga struktur API menjadi lebih modular dan mudah dikelola.

Gambar 3. 16 Gambar Potongan Kode Contoh Endpoint Module (Sumber: Hasil Olahan Sendiri)

Pada modul kalender, setiap *endpoint* didefinisikan dengan metode HTTP yang spesifik, seperti **GET** untuk mengambil data, **POST** untuk membuat data baru, serta **PUT** dan **DELETE** untuk memperbarui dan menghapus data. Setiap rute juga dilindungi oleh *middleware* authenticateToken untuk memastikan keamanan akses hanya bagi pengguna yang memiliki hak akses valid.

3.8 Pengujian dan Pembahasan

Pengujian sistem dilakukan untuk memastikan bahwa setiap fitur backend yang dikembangkan telah berjalan sesuai dengan kebutuhan sistem dan spesifikasi yang telah ditentukan. Metode pengujian yang digunakan pada penelitian ini adalah **Black Box Testing**, yaitu metode pengujian yang berfokus pada pemeriksaan fungsionalitas sistem berdasarkan *input* dan *output* tanpa melihat *struktur* kode program di dalamnya.

Pengujian dilakukan terhadap seluruh endpoint API backend menggunakan *tools* **Postman**. Setiap endpoint diuji dengan berbagai skenario *input*, baik *input* valid maupun *input* tidak valid, untuk memastikan sistem dapat memberikan respon yang sesuai, baik dalam kondisi berhasil maupun gagal.

3.8.1 Skenario Pengujian (*Black box testing*)

Metode pengujian yang digunakan dalam penelitian ini adalah **Black Box Testing**, dengan tujuan untuk menguji kesesuaian antara *input* yang diberikan dengan *output* yang dihasilkan oleh sistem.

Pengujian dilakukan dengan *cara*:

1. Mengirimkan *request ke* endpoint API menggunakan Postman.
2. Memberikan *input* sesuai dengan skenario pengujian (valid dan tidak valid).
3. Mengamati respon yang dihasilkan oleh sistem.
4. Membandingkan hasil aktual dengan hasil yang diharapkan.

Pengujian ini difokuskan pada:

- Validasi *input* pengguna
- Autentikasi dan otorisasi
- Keberhasilan proses CRUD
- Penanganan error oleh sistem

3.8.2 Hasil Pengujian

Hasil pengujian diperoleh setelah seluruh skenario pengujian Black Box Testing dijalankan terhadap setiap endpoint backend yang dikembangkan. Pengujian

dilakukan dengan mengirimkan *request ke* API menggunakan aplikasi Postman dan mengamati respon yang dihasilkan oleh sistem berdasarkan *input* yang diberikan.

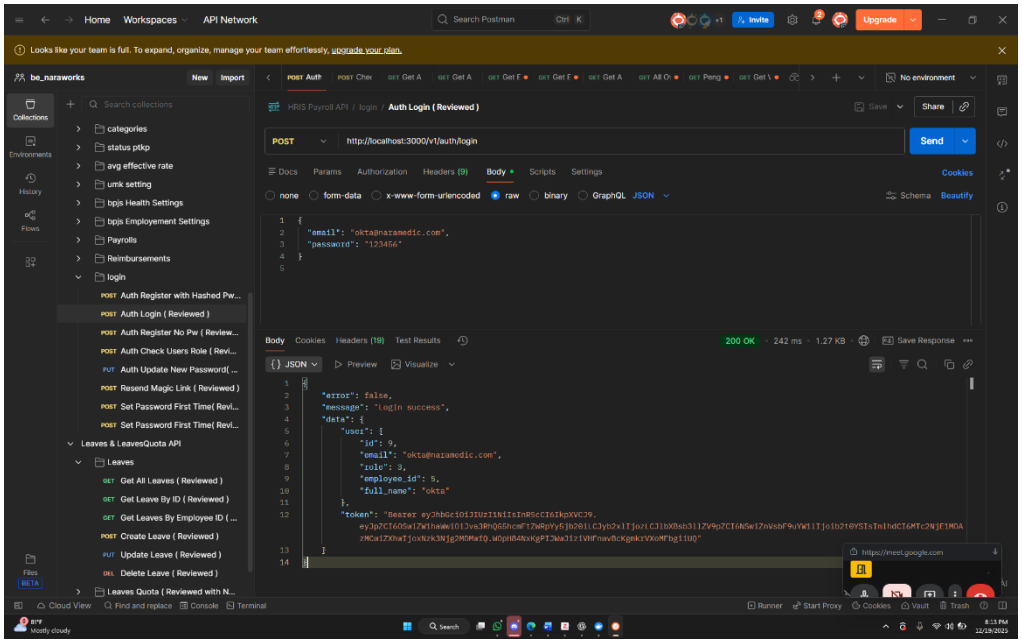
Hasil pengujian disajikan dalam bentuk tabel untuk mempermudah analisis dan evaluasi terhadap fungsionalitas sistem. Tabel tersebut menampilkan informasi mengenai fitur yang diuji, skenario pengujian, hasil yang diharapkan, hasil aktual yang diperoleh dari sistem, serta status keberhasilan pengujian.

Selain tabel hasil pengujian, pada bagian ini juga disertakan **screenshot respon API dari Postman** sebagai bukti bahwa pengujian telah dilakukan. Screenshot tersebut menunjukkan respon sistem dalam kondisi berhasil maupun gagal, termasuk kode status *HTTP* dan pesan respon yang dihasilkan oleh sistem. Bukti pengujian ini digunakan untuk memperkuat kesesuaian antara hasil aktual dengan hasil yang diharapkan.

Berdasarkan data hasil pengujian yang ditampilkan pada tabel dan didukung oleh bukti respon Postman, dapat dianalisis bahwa sebagian besar fitur backend telah berjalan sesuai dengan kebutuhan sistem dan mampu menangani berbagai skenario *input* dengan baik.

Sebagai bukti implementasi dari skenario pengujian yang telah dirancang, berikut adalah penjelasan detail mengenai salah satu pengujian fungsional utama, yaitu fitur autentikasi (*Login*), yang dilakukan menggunakan alat bantu Postman:

a. **Bukti Pengujian Autentikasi (*Login User*)** Fitur ini diuji untuk memastikan bahwa sistem hanya memberikan akses kepada pengguna yang memiliki kredensial valid. Proses pengujian ini mencakup pengiriman data email dan kata sandi ke *endpoint* /v1/auth/login. Hasil aktual dari pengujian ini dapat dilihat pada Gambar 3.10.



Gambar 3.17 Gambar Potongan Kode Pengujian Login (Sumber: Hasil Olahan Sendiri)

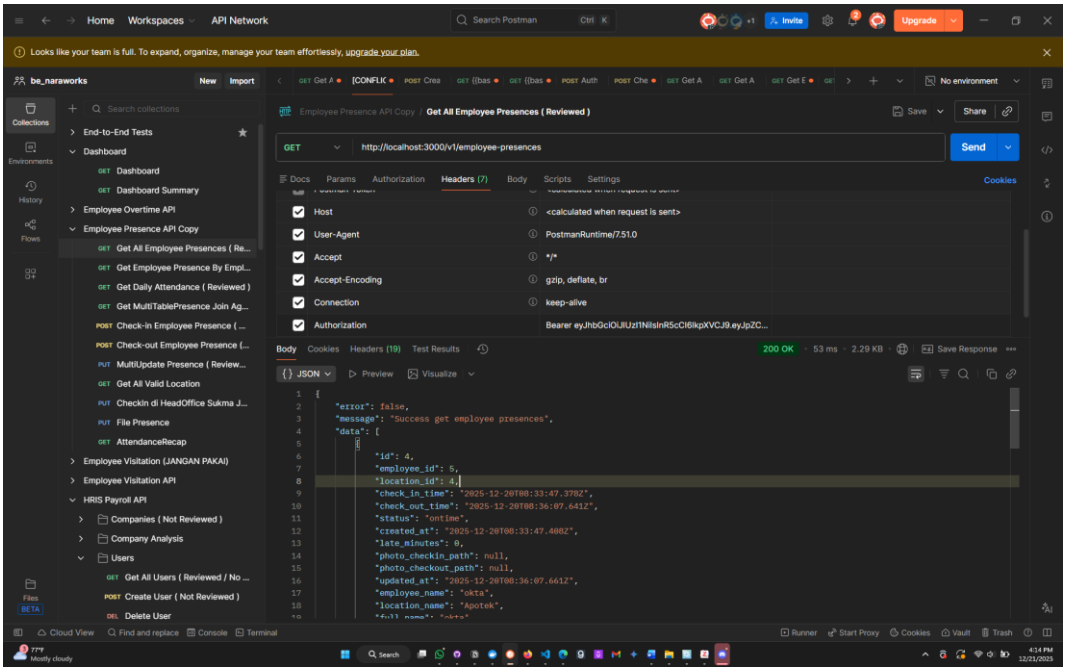
Berdasarkan Gambar 3.10 di atas, dapat dilakukan analisis terhadap komponen respon sistem sebagai berikut:

1) **Metode dan URL**: Pengujian dilakukan menggunakan metode POST ke alamat *host* lokal dengan *path* /v1/auth/login.
2) **Struktur *Request***: Data dikirimkan dalam format JSON yang berisi atribut email dan password.
3) **Kode Status HTTP**: Sistem mengembalikan status **200 OK**, yang menandakan bahwa *request* berhasil diproses oleh *server*.
4) **Konten Respon**: Respon JSON menunjukkan bahwa atribut error bernilai false dengan pesan "Login success".
5) **Data dan Token**: Sistem berhasil mengembalikan objek user yang berisi informasi profil pengguna (seperti email, role, dan full_name) serta menyertakan token berbasis JWT (*JSON Web Token*). Token ini nantinya akan digunakan oleh *frontend* sebagai autentikasi pada *header* untuk mengakses *endpoint* lain yang dilindungi.

b. Bukti Pengujian Pengambilan Data Kehadiran (Get All Employee Presences)

Setelah pengguna berhasil melakukan autentikasi dan mendapatkan access token, pengujian selanjutnya adalah memastikan bahwa token tersebut valid

untuk digunakan dalam melakukan permintaan data ke endpoint yang bersifat terbatas (protected routes). Pengujian dilakukan pada endpoint /v1/employee-presences menggunakan metode GET melalui aplikasi Postman, dengan hasil yang ditunjukkan pada Gambar 3.11.



Gambar 3.18 Gambar Tanda Bukti Pengujian Endpoint pada Postman (Sumber: Hasil Olahan Sendiri)

1) Berdasarkan hasil pengujian pada Gambar 3.11, berikut adalah analisis teknis terhadap respon sistem:
2) Mekanisme Autentikasi: Pengujian ini menyertakan header Authorization dengan tipe Bearer Token yang berisi nilai JWT hasil dari proses login sebelumnya. Tanpa token ini, sistem secara otomatis akan menolak akses dengan kode status 401 Unauthorized.
3) Kode Status HTTP: Sistem mengembalikan status 200 OK, yang mengonfirmasi bahwa token valid dan pengguna memiliki hak akses untuk menarik data riwayat kehadiran.
4) Struktur Data Respon: Respon JSON menunjukkan atribut error: false dan pesan "Success get employee presences". Data yang dikembalikan berupa array objek yang memuat informasi detail kehadiran, seperti:

5) check_in_time dan check_out_time: Waktu presisi karyawan melakukan absensi.
6) location_name: Nama lokasi tempat absensi dilakukan (contoh: "Apotek").
7) status: Keterangan ketepatan waktu (contoh: "ontime").
8) Integritas Data: Data yang ditampilkan mencakup informasi profil lengkap seperti employee_name dan full_name, yang membuktikan bahwa backend berhasil melakukan proses join antar tabel (tabel presences dan tabel employees) di lapisan Data Tier.

Tabel 3.7 Hasil Pengujian Fitur *Backend*

No	Fitur	Skenario	Hasil Diharapkan	Hasil Aktual	Status
1	Login User	User memasukkan email dan *password* valid	Sistem mengembalikan token JWT dan status 200	Token JWT berhasil dikembalikan	Berhasil
2	Login User	User memasukkan *password* salah	Sistem menolak login dan menampilkan pesan error	Respon error “Invalid credentials”	Berhasil
3	Login User	Email tidak terdaftar	Sistem menampilkan pesan user tidak ditemukan	Respon error “User not found”	Berhasil
4	Get All Users	Request dengan token valid	Sistem menampilkan daftar seluruh user	Data user berhasil ditampilkan	Berhasil
5	Get All Users	Request tanpa token	Sistem menolak akses	Respon error unauthorized (401)	Berhasil

6	Create User	Data user valid	Data user berhasil disimpan	User berhasil ditambahkan	Berhasil
7	Create User	Data tidak lengkap	Sistem menolak *request*	Respon error validasi	Berhasil
8	Update User	ID user valid	Data user berhasil diperbarui	Data berhasil diupdate	Berhasil
9	Update User	ID user tidak ditemukan	Sistem menampilkan error	Respon “User not found”	Berhasil
10	Delete User	ID user valid	Data user berhasil dihapus	Data user terhapus	Berhasil
11	Delete User	ID user tidak ditemukan	Sistem menolak penghapusan	Respon error	

(Sumber: Hasil Olahan Sendiri)

Berdasarkan hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa seluruh endpoint backend yang dikembangkan telah berjalan sesuai dengan kebutuhan sistem. Setiap fitur mampu menangani *input* valid maupun *input* tidak valid dengan baik, serta memberikan respon yang sesuai.

Namun, selama proses pengujian terdapat beberapa kendala, antara lain:

1. **Kesalahan *form*at *request***, seperti header authorization yang tidak disertakan, menyebabkan respon unauthorized.
2. **Validasi *input* belum sepenuhnya** konsisten pada beberapa endpoint sehingga diperlukan penyesuaian tambahan pada sisi backend.
3. **Pengelolaan error message** perlu distandarisasi agar lebih mudah dipahami oleh *frontend developer*.

Kendala-kendala tersebut telah diperbaiki selama proses pengembangan sehingga sistem dapat berjalan secara optimal.

3.8.3 Kendala Pengembangan

Selama proses pengembangan sistem *backend*, terdapat beberapa kendala teknis yang dihadapi baik kendala teknis dan non teknis, kendala kendala tersebut antara lain :

1. Kendala Teknis
   a. Isu konsistensi data kehadiran
      Terjadi ketidaksesuaian data antara waktu *check*-in, *check*-out, dan status lembur akibat proses *up*date data yang melibatkan beberapa tabel secara bersamaan (presence, overtime, dan visitation).
   b. *Bug* pada perhitungan waktu kerja dan lembur
      Perhitungan durasi kerja dan lembur mengalami ketidaktepatan karena perbedaan *form*at waktu dan pengelolaan timezone antara *server* dan *database*.
   c. Validasi alur bisnis yang kompleks
      Ditemukan kendala dalam menerapkan aturan bisnis, seperti pembatasan *check*-in lembur hanya jika pengajuan lembur telah disetujui, yang membutuhkan pengecekan status approval sebelum proses kehadiran.
   d. Kesulitan integrasi autentikasi dan otorisasi
      Implementasi *JWT* dan role-based access control memerlukan penyesuaian pada *middleware* agar setiap *endpoint* hanya dapat diakses sesuai hak pengguna.
   e. Optimasi per*form*a *query database*
      Endpoint rekap kehadiran harian yang melibatkan join beberapa tabel memiliki waktu respon yang cuk*up* tinggi sebelum dilakukan optimasi *query* dan indexing.
   f. Kurangnya dokumentasi *endpoint API* dan *API* contract pada tahap awal
      Pada awal pengembangan, dokumentasi terkait *endpoint API*, *struktur request*–response, serta *API* contract belum tersedia secara jelas. Hal ini menyebabkan kebingungan dalam menentukan alur pembuatan *endpoint*, parameter yang digunakan, serta konsistensi *form*at data antar modul.
   g. Ketidakkonsistenan *struktur API* antar modul

Akibat belum adanya *API* contract yang baku sejak awal, terdapat perbedaan pola penamaan *endpoint*, *struktur* payload, dan response antar modul seperti presence, overtime, dan leave, sehingga memerlukan penyesuaian ulang pada tahap integrasi.

2. Kendala Non Teknis
    a. Perubahan kebutuhan sistem secara bertahap

       Beberapa kebutuhan fungsional mengalami penyesuaian selama proses pengembangan, terutama pada modul kehadiran, lembur, dan cuti. Terkadang pembatasan tertentu yang harus dilakukan oleh sistem terkait pembatasan akses fitur tertentu belum dilakukan di alur bisnis dan alur kerja perusahaan sebelumnya.
    b. Keterbatasan dokumentasi awal sistem

       Tidak semua alur bisnis terdokumentasi secara rinci di awal, sehingga diperlukan pemahaman tambahan melalui diskusi dan eksplorasi kode yang sudah ada. Hal hal terkait alur dan peraturan yang belum dilaksanakan sebelumnya pada perusahaan pun terkadang ikut menjadi requirement tambahan seperti halnya pengajuan cuti, sehingga tim harus melakukan brainstorming terhadap alur yang belum pasti.
    c. Koordinasi dan komunikasi tim

       Proses sinkronisasi pemahaman antar anggota tim *backend* dan pihak terkait termasuk *frontend* dan stakeholder membutuhkan waktu, terutama dalam menyamakan interpretasi aturan bisnis sistem HR.

3.8.4 Solusi dan Pemecahan Masalah

3.8.4.1 Solusi atas kendala teknis

Untuk mengatasi kendala teknis yang muncul, langkah langkah berikut dilakukan :

a. Refactoring kode dan pemisahan logika bisnis

   Logika perhitungan waktu, validasi kehadiran, dan approval dipisahkan *ke* dalam layer service agar lebih terstruktur dan mudah diuji.
b. Penyesuaian perhitungan waktu dan timezone

Seluruh pencatatan waktu diseragamkan menggunakan waktu *server* dan timezone yang sama antara aplikasi dan *database* untuk menghindari perbedaan hasil perhitungan.

c. Penambahan validasi berbasis status

Sistem ditambahkan validasi untuk memastikan bahwa proses *check*-in lembur hanya dapat dilakukan jika status pengajuan lembur telah disetujui.

d. Penyempurnaan *middleware* autentikasi dan otorisasi

Middleware JWT diperbaiki agar dapat memverifikasi token, role pengguna, dan membatasi akses *endpoint* sesuai kewenangan.

e. Optimasi *query database*

Dilakukan optimasi *query* dengan penggunaan indexing, pagination, dan pengurangan join yang tidak diperlukan untuk meningkatkan per*forma* *API*.

3.8.4.2 Solusi atas kendala non-teknis

Untuk mengatasi kendalanon teknis, dilakukan beberapa pendekatan berikut:

a. Diskusi dan klarifikasi kebutuhan dengan mentor atau tim terkait

Setiap perubahan kebutuhan dibahas terlebih dahulu untuk memastikan implementasi sesuai dengan tujuan sistem.

b. Mempelajari dokumentasi resmi dan referensi teknis

Dokumentasi resmi *framework*, *library*, serta referensi best practice digunakan sebagai acuan dalam menyelesaikan permasalahan teknis.

c. Pencatatan perubahan dan evaluasi berkala

Setiap perubahan penting dicatat dan dievaluasi agar tidak menimbulkan konflik dengan fitur yang telah berjalan.

BAB IV

PENUTUP

4.1 Kesimpulan

Berdasarkan hasil perancangan dan implementasi sistem yang telah dilakukan, dapat disimpulkan bahwa sistem *Human Resource Management* (HRM) dan *payroll* yang dikembangkan mampu mendukung proses manajemen karyawan, absensi, serta penggajian secara terintegrasi. Sistem ini dirancang untuk mempermudah pengelolaan data karyawan dan meningkatkan efisiensi proses administrasi yang sebelumnya dilakukan secara manual. Perancangan *database* telah dilakukan dengan *struktur* tabel yang jelas dan terorganisir, mencakup entitas utama seperti *employees*, *users*, dan *presence_location*, sehingga mendukung penyimpanan dan pengelolaan data secara sistematis.

Selain itu, alur logika sistem yang digambarkan melalui flowchart menunjukkan proses validasi absensi yang terstruktur, dimulai dari proses *input* data oleh pengguna, validasi lokasi dan waktu, hingga pemberian respon sistem. *API* contract yang dirancang juga telah mendukung komunikasi antara *frontend* dan *backend* melalui *endpoint* yang sesuai dengan kebutuhan sistem, sehingga aplikasi dapat diakses dan dikembangkan secara fleksibel. Dari sisi infrastruktur, perangkat keras yang digunakan, yaitu laptop pengembang dan VPS sebagai media *hosting*, telah memenuhi kebutuhan pengembangan, pengujian, hingga publikasi sistem.

4.2 Saran

Untuk pengembangan sistem *ke* depannya, terdapat beberapa saran yang dapat dipertimbangkan. Dari sisi *database*, perlu dilakukan optimasi dengan normalisasi lebih lanjut serta penambahan indeks pada kolom yang sering digunakan dalam proses *query* agar per*forma* sistem dapat meningkat. Dari aspek keamanan, sistem disarankan untuk menerapkan enkripsi pada data sensitif seperti *password*, NIK,

dan NPWP, serta menggunakan mekanisme autentikasi berbasis token seperti *JWT* atau OAuth guna meningkatkan perlindungan data pengguna.

Selain itu, dalam rangka mendukung skalabilitas sistem, *monitoring* beban VPS perlu dilakukan secara berkala dan disertai dengan perencanaan *up*grade *server* apabila jumlah pengguna meningkat. Pengembangan *API* juga dapat ditingkatkan dengan menambahkan dokumentasi *API* menggunakan *tools* seperti *Swagger* atau *OpenAPI* agar proses integrasi dengan sistem lain menjadi lebih mudah dan terstruktur. Dari sisi pengalaman pengguna, perbaikan pada tampilan antarmuka serta pemberian notifikasi yang jelas pada setiap proses absensi atau pengajuan cuti sangat disarankan. Terakhir, untuk menjamin kualitas dan keandalan sistem, perlu dilakukan pengujian secara menyeluruh yang mencakup unit *testing*, integration *testing*, dan *user acceptance testing* sebelum sistem digunakan secara penuh.

DAFTAR PUSTAKA

Alhilali, A. et al. 2019. "Multi-Objective Attendance And Management Information

System Using Computer Application In Industry Strip". Indonesian Journal of Electrical Engineering and Computer Science, 19, 371–381. https://doi.org/10.11591/ijeecs.v16.i1.pp371-381.

Anonim. Agile Manifesto Indonesia. (n.d.). https://agilemanifesto.org/iso/id/manifesto.html. [24 November 2025]

Anonim. The Express Handbook. (n.d.). https://www.dbooks.org/the-express-handbook-5614276129/pdf/. [22 November 2025]

Asana. 2025. What Is Agile Methodology?. https://asana.com/resources/agile-methodology. [24 November 2025]

Dessler, G. (n.d.). Human Resource Management (15th ed.). https://studylib.net/doc/27769152/812438111-human-resource-management-15th-edition-textbook. [22 November 2025]

Kabul, E. R. 2024. "Penggunaan Teknologi HRM (Human Resource Management) Untuk Meningkatkan Efisiensi Dan Efektivitas Manajemen Sumber Daya Manusia". Blantika: Multidisciplinary Journal, 2(4), 421–429.

Laudon, K. C. dan Laudon, J. P. 2022. Management Information Systems: Managing The Digital Firm (17th ed.). Hoboken: Pearson.

Martin, R. C. 2014. Agile Software Development: Principles, Patterns, and Practices. https://github.com/y-t99/agile-development/blob/main/99-Agile%20Software%20Development%20Principles%2C%20Patterns%2C%20and%20Practices%20(Robert%20C.%20Martin).pdf. [24 November 2025]

Node.js. (n.d.). Introduction To Node.js. https://nodejs.org/en/learn/getting-started/introduction-to-nodejs. [22 November 2025]

Pertiwi, N. E. C. P. dan Zagladi, A. N. 2025. "Analisis Penggunaan Sistem Integrasi PT Angkasa Raya Steel (SISTAR) Dalam Peningkatan Efektivitas Pengelolaan SDM Pada Departemen HRD". Jurnal Ilmu Ekonomi, Manajemen Dan Bisnis, 3(2), 70–81.

PostgreSQL Documentation. (n.d.). PostgreSQL 16.0 Documentation. https://www.postgresql.org/docs/16/index.html. [20 Desember 2025]

Purnama, S., Kamal, M. dan Yadila, A. B. 2023. "Application Of RESTful Method With JWT Security And Haversine Algorithm On Web Service-Based

Teacher Attendance System". International Transactions on Artificial Intelligence, 2(1), 33–39. https://doi.org/10.33050/italic.v2i1.400.

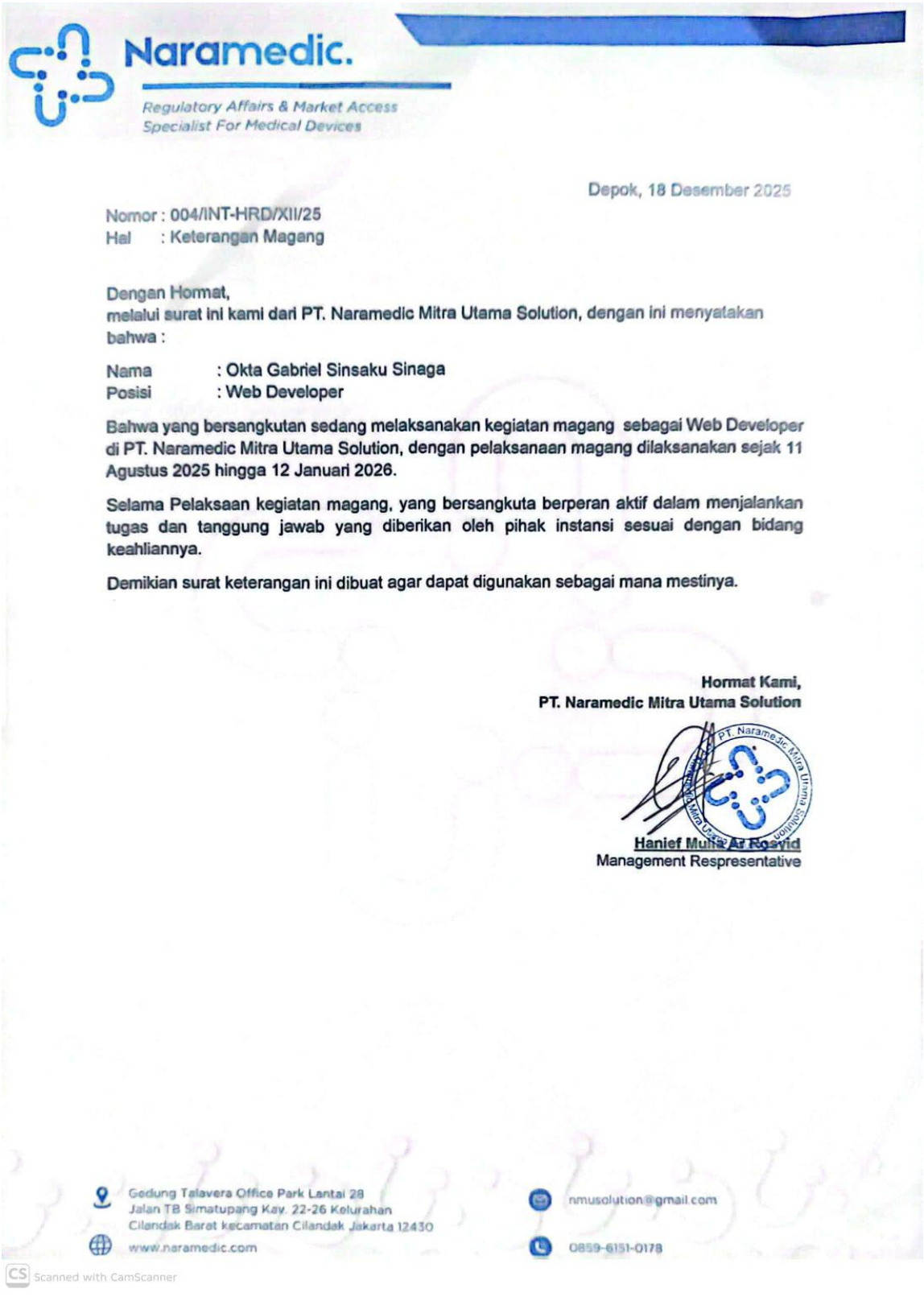
Scrum Guide. (n.d.). Scrum Guide | Scrum Guides. https://scrumguides.org/scrum-guide.html. [5 Desember 2025]

Susana, R. dan Lukman, H. 2023. "Measurement Of Acceptance Of Online Attendance System With Technology Acceptance Model Approach: Case In Directorate General Of Christian Community, Ministry Of Religious Affairs Of The Republic Indonesia". International Journal Of Application On Economics And Business (IJAEB), 1(2), 36–44.

Turnbull, J. 2017. The Docker Book (V17.03.0). https://github.com/dskcode/docker-books/blob/master/The%20Docker%20Book%20-%20James%20Turnbull%20-%20v17.03.0.pdf. [23 November 2025]

LAMPIRAN

Lampiran 1 Surat Keterangan Masih Melaksanakan Magang



Naramedic.
Regulatory Affairs & Market Access
Specialist For Medical Devices

Depok, 18 Desember 2025

Nomor : 004/INT-HRD/XII/25
Hal : Keterangan Magang

Dengan Hormat,
melalui surat ini kami dari PT. Naramedic Mitra Utama Solution, dengan ini menyatakan bahwa :

Nama : Okta Gabriel Sinsaku Sinaga
Posisi : Web Developer

Bahwa yang bersangkutan sedang melaksanakan kegiatan magang sebagai Web Developer di PT. Naramedic Mitra Utama Solution, dengan pelaksanaan magang dilaksanakan sejak 11 Agustus 2025 hingga 12 Januari 2026.

Selama Pelaksaan kegiatan magang, yang bersangkuta berperan aktif dalam menjalankan tugas dan tanggung jawab yang diberikan oleh pihak instansi sesuai dengan bidang keahliannya.

Demikian surat keterangan ini dibuat agar dapat digunakan sebagai mana mestinya.

Hormat Kami,
PT. Naramedic Mitra Utama Solution

Hanief Muha Ar Rosyid
Management Respresentative

Gedung Talavera Office Park Lantai 28
Jalan TB Simatupang Kav. 22-26 Kelurahan
Cilandak Barat kecamatan Cilandak Jakarta 12430
www.naramedic.com
nmusolution@gmail.com
0859-6151-0178

Scanned with CamScanner

Lampiran 2 Logbook Aktivitas Praktik Kerja Lapangan

KEMENTERIAN PENDIDIKAN TINGGI,
RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI JAKARTA
JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER
Jl. Prof. DR. G.A. Siwabessy, Kampus UI, Depok 16425
Telp: (021)91274097, Fax : (021) 7863531, (021)7270036 Hunting
Laman :http://www.pnj.ac.id, e-mail : tik.pnj@gmail.com

F3

BUKU PENGHUBUNG
PEMBIMBING MAGANG INDUSTRI

1. Nama perusahaan/ Industri : PT Naramedic Mitra Utama Solution
2. Alamat : Blok C, Jl. Kemang Swatama No.27, Kalibaru, Kec. Cilodong, Kota Depok, Jawa Barat 16473
3. Judul Magang : Pengembangan Modul Backend Presensi dan Human Resources Management Terintegrasi dengan Sistem Penggajian
4. Nama Pembimbing Industri : Hanief Mulia Ar Rasyid

No	Hari/Tgl	Aktivitas yang Dilakukan	Tandatangan
1	12-16 Aug 2025	Orientasi magang, pengenalan perusahaan, pemahaman alur bisnis HR & presensi, serta studi awal sistem yang berjalan.	
2	18-23 Aug 2025	Analisis kebutuhan sistem (user requirement) untuk modul presensi dan HRM serta diskusi teknis dengan pembimbing industri.	
3	25-30 Aug 2025	Perancangan arsitektur backend, struktur database, dan relasi tabel untuk modul employee data, presence, dan overtime.	
4	1-6 Sep 2025	Implementasi modul Authentication & Authorization pada backend menggunakan role-based access control.	
5	08-13 Sep 2025	Pengembangan modul Employee Data (CRUD) dan integrasi dengan database utama.	
6	15-20 Sep 2025	Pengembangan Presence Module untuk pencatatan kehadiran karyawan (check-in, check-out, dan rekap presensi).	
7	22-27 Sep 2025	Pengembangan Overtime Module meliputi pengajuan, persetujuan, dan pencatatan data lembur karyawan.	
8	29 Sep - 04 Oct 2025	Pengembangan Leaves Module untuk pengajuan cuti, persetujuan, serta	

(lanjutan)

		pencatatan sisa cuti karyawan.	
9	06-11 Oct 2025	Pengembangan Visitation Module untuk pencatatan kunjungan kerja karyawan ke klien atau lokasi tertentu.	
10	13-18 Oct 2025	Pengembangan Company Major Module (departemen, jabatan, dan struktur organisasi).	
11	20-25 Oct 2025	Integrasi antar modul backend serta penyesuaian business logic sesuai kebijakan perusahaan.	
12	27 Oct - 01 Nov 2025	Pengembangan dan dokumentasi API backend untuk integrasi dengan aplikasi frontend.	
13	03-08 Nov 2025	Integrasi backend dengan frontend serta pengujian fungsionalitas modul presensi dan HRM.	
14	10-15 Nov 2025	Implementasi integrasi data presensi, lembur, dan cuti dengan sistem penggajian (tanpa perhitungan payroll).	
15	17-22 Nov 2025	Pengujian sistem menyeluruh, perbaikan bug, dan optimasi performa backend.	
16	24-29 Nov 2025	Pengujian sistem menyeluruh, perbaikan bug, dan optimasi performa backend.	
17	01-06 Dec 2025	Penyusunan dokumen laporan magang dan bimbingan kepada dosen pembimbing	
18	08-13 Dec 2025	Finalisasi dokumen laporan magang dan pemenuhan kuota minimal bimbingan.	
19	15-19 Dec 2025	Deployment sistem backend ke environment server perusahaan dan konfigurasi environment.	

Depok, 6 Desember 2025

Management Representative,

Hanief Mulia Ar Rosyid

Lampiran 3 User Requirement

/

KEMENTERIAN PENDIDIKAN TINGGI,
RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI JAKARTA
JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER
Jl. Prof. DR. G.A. Siwabessy, Kampus UI, Depok 16425
Telp: (021)91274097, Fax : (021) 7863531, (021)7270036 Hunting
Laman :http://www.pnj.ac.id, e-mail : tik.pnj@gmail.com

F2

USER REQUIREMENT

(Kepentingan Pengguna/Perusahaan)

Nama Pembimbing Industri : Hanief Mulia Ar Rosyid
Bagian/Departemen : Web Development / IT Support

No.	Modul/Unit yang dikerjakan	*User Requirement*/Spesifikasi	Paraf (Pembimbing Industri)
1.	Authentication & Authorization	Sistem menyediakan mekanisme autentikasi pengguna (login dan logout) serta otorisasi berbasis role untuk membatasi akses fitur sesuai dengan hak pengguna (Admin, HR, Manager, Employee).	
2.	Employee Data	Sistem mampu mengelola data karyawan secara terpusat (CRUD) meliputi identitas karyawan, jabatan, departemen, status kerja, dan informasi pendukung lainnya.	
3.	Presence Module	Sistem mencatat dan mengelola data kehadiran karyawan, termasuk jam masuk, jam pulang, dan rekap kehadiran per periode sebagai data operasional perusahaan.	
4.	Visitation Module	Sistem mengelola data kunjungan kerja karyawan (visitation) ke lokasi atau klien tertentu, termasuk tanggal, tujuan kunjungan, dan keterangan aktivitas.	
5.	Overtime Module	Sistem mengelola pengajuan, persetujuan, dan pencatatan lembur karyawan serta menyimpan data lembur secara terstruktur dan terdokumentasi.	

(lanjutan)

6.	Company Major Module	Sistem menyediakan modul untuk pengelolaan data utama perusahaan seperti struktur organisasi, departemen, jabatan, dan kebijakan internal yang digunakan oleh modul lain.	
7.	Integration with Frontend	Sistem backend menyediakan API yang terstruktur dan terdokumentasi untuk diintegrasikan dengan aplikasi frontend sehingga proses pertukaran data berjalan dengan aman dan konsisten.	
8.	Leaves Module	Sistem mengelola pengajuan cuti karyawan, proses persetujuan cuti, serta pencatatan sisa cuti berdasarkan kebijakan perusahaan.	
9.	Integration with Payroll	Sistem menyediakan integrasi data yang dibutuhkan oleh sistem payroll, seperti kehadiran, lembur, dan cuti, tanpa melakukan perhitungan payroll secara langsung.	
10.	Deployment Bug Fixing	Sistem dideploy ke lingkungan server yang ditentukan serta dilakukan perbaikan bug dan penyesuaian sistem berdasarkan hasil pengujian dan umpan balik pengguna.	

Depok, 6 Desember 2025
Pembimbing Industri,

Hanief Fatulloh Ar Rosyid

Lampiran 4 Dokumentasi Kegiatan Praktik Kerja Lapangan